VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Zpracování velkých objemů nestrukturovaných dat na platformě Hadoop

Processing Large Volumes of Unstructured Data on Hadoop Platform

2015 Martin Prouza

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Prouza**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Zpracování velkých objemů nestrukturovaných dat na platformě Hadoop
Processing Large Volumes of Unstructured Data on Hadoop Platform

Zásady pro vypracování:

Cílem práce je na základě zkušeností ze zpracovávaného tématu zhodnotit možnosti využití Hadoop platformy při zpracování velkého objemu různorodých dat.

1. Nastudujte NoSQL databázové systémy, zaměřte se na platformu Hadoop.
2. Proveďte analýzu velkého objemu nestrukturovaných dat, např. obsahu logů, pomocí Hadoop.
3. Vyhodnoťte výsledky experimentů.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Jiří Skácelík**

Konzultant bakalářské práce: doc. Ing. Michal Krátký, Ph.D.

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení Studenta

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

„Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava."

V Ostravě Dne 7.5.2015 Podpis .....

Poděkování

Rád bych poděkoval všem, kteří mi poskytli odbornou pomoc, cenné rady a materiály, které jsem využil k napsání této práce.

V Ostravě dne 7.5.2015 .....

podpis

Abstrakt

Tato práce se zabývá zpracováním nestrukturovaného textu na platformě Hadoop. První část se zaměřuje na důvody vzniku konceptu Big Data. Vysvětlím problematiku dat dnešní doby a ukážu, proč jsou běžné databázové systémy nevhodné pro práci s nestrukturovanými či velkými objemy dat.

Další část je zaměřená na teorii konceptu Big Data a jeho zpracování na platformě Hadoop. Představím Hadoop architekturu, a jak se liší od běžného databázového skladu. Také vysvětlím teorii paralelního zpracování dat, a jak je toto paralelní zpracování řešeno na platformě Hadoop.

Poslední část se zabývá praktickou částí řešení zpracování obsahu nestrukturovaných serverových logů na platformě Hadoop. Jakým způsobem jsme schopni tato data analyzovat a získat z nich využitelné informace. Výsledek této části budu prezentovat v reportingovém programu QlikView.

Výsledek byl rovněž zpracován na klasické SQL databázi, aby se porovnal přínos Hadoop platformy při zpracování nestrukturovaných dat.

Klíčová slova

Distribuovaný souborový systém, Hadoop, HDFS, MapReduce, nestrukturovaná data, NoSQL, paralelní zpracování dat, QlikView, ngram, Hive

Abstract

This thesis is concerned with processing unstructured text on Hadoop platform. First part focuses on the reasons of creation Big Data concept. I explain data issue of these days and show, why common database systems are inappropriate for working with huge amounts of unstructured data.

The next part focuses on theory about Big Data concept and processing on Hadoop platform. I introduce Hadoop architecture and how it differs from common warehouse database. I also explain parallel data processing theory and how parallel processing is solved on Hadoop platform.

The last part focuses on practical part to solve processing huge amounts of unstructured server logs on Hadoop platform. How can we analyze these data and get some valuable information from them. The result will be presented in reporting program QlikView.

Result was also processed on classic SQL database, to compare Hadoop platform contribution in processing unstructured data.

Keywords

Distributed file system, Hadoop, HDFS, MapReduce, NoSQL, parallel data processing, QlikView, unstructured data, ngram, Hive

Obsah

1 Úvod

Každý stroj, automaticky řízený počítačem nebo člověkem, nám generuje data a jsou různých typů a velikostí. Pokud dokážeme ovládnout tyto data, může nám to pomoci v mnoha směrech vývoje. Z těchto dat můžeme vyhledat, kde firma přichází o peníze nebo podle trendů minulých měsíců či let předcházet problémům, které dříve nastaly za stejných či podobných podmínek. Ten kdo tyto informace z dat dokáže získat, bude mít pokaždé o krok napřed před ostatními. Proč to ale bývá takový problém? Když máme svá data, tak se na ně koukneme a zjistíme z nich vše, co potřebujeme. To však není úplně jednoduché. Je rozdíl mezi daty a informacemi. Tento základní pojem se musí umět rozlišit. Data sama o sobě mají nulovou informativní hodnotu a teprve analýzou se z nich mohou dostat informace a výsledné znalosti.

Kolik ale takových dat jsme schopni analyzovat? Jedním si můžeme být jistí. A to tím, že těchto dat budeme mít stále více, protože naše stroje nám stále produkují nové a nové data, která musíme někam ukládat. Tím se ale dostáváme do dalších problémů kam tyto data skladovat. Úložiště, které nám stačí dnes, už zítra stačit nemusí. A budeme se muset začít poohlížet po nějakém větším úložišti dat.

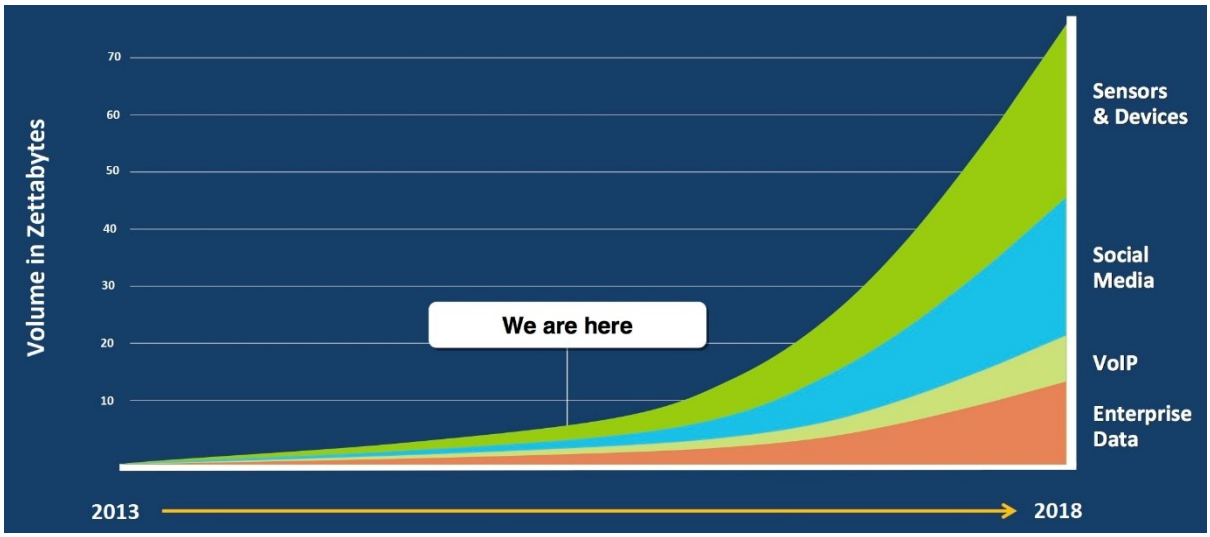
1.1 Cíle

V mé práci se pokusím objasnit co nejvíce otázek, které směřují k problematice ukládání a zpracování velkých objemů nestrukturovaných dat na Hadoop platformě. Vysvětlení konceptu Big Data a v čem se liší oproti běžným databázovým systémům. Nakonec ukáži, jakých výsledků pomoci této platformy jsme schopni dosáhnout.

2 Vývoj dat v historii

2.1 Velikost dat podle doby

V různých letech naší historie byly různé nároky na skladování dat. Jednoznačně ale můžeme říci, že objem dat, která je nutno uložit rok od roku stoupá. Jaký byl ale trend přibývání dat na světě? Existuje mnoho firem, které monitorují objemy produkovaných dat. Tyto hodnoty nemohou být sice přesné, ale získáme tím hlavní představu, v jakých soustavách se pohybujeme. Obzvláště jaký je trend příbytku dat v současných letech a jejich predikce do let následujících [1].



Obrázek 1 - Vývoj dat [1]

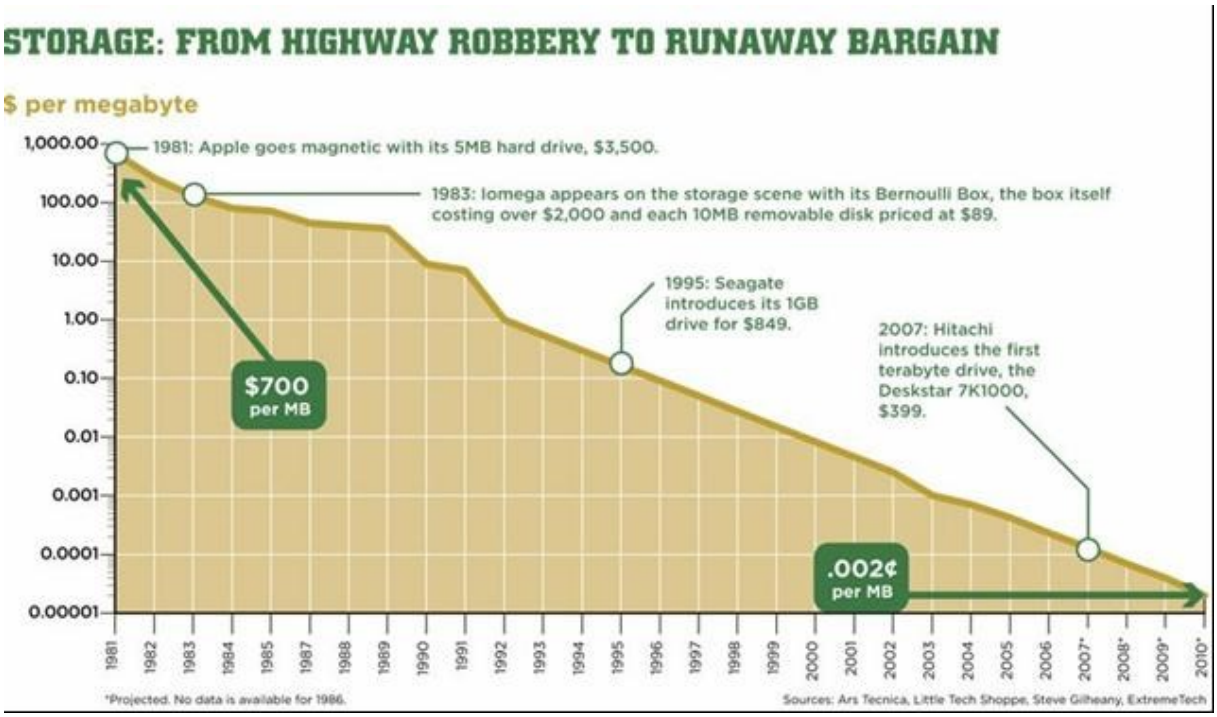
Z obrázku (viz Obrázek 1 - Vývoj dat lze snadno zjistit, že trend příbytku dat je podobný exponenciální křivce, z čehož jde vyvodit, že okolo 80% všech současných dat bylo vygenerováno v posledních dvou letech. A tento trend nemá ustávat ani v letech následujících. Pokud tedy bude současný trend příbytku objemu dat narůstat i v příštích letech, tak nároky na skladování dat budou za dva roky 5krát vyšší a v dalších dvou letech, tedy příštích čtyřech letech, budou nároky na skladování dat 25krát vyšší. Důvodů tohoto nárůstu dat je několik. Nejvíce dat nám v současné době pochází z monitorovacích zařízení a senzorů. Rozeberme si třeba senzory používané v letecké dopravě. Letadlo při letu generuje kolem miliardy řádků kódu z jednoho motoru za asi půl hodiny [2]. Pokud to tedy přepočítáme na běžné měřící jednotky, které se používají v informatice, takové množství řádků nám bude odpovídat asi 10 TB dat, která se musí někde uchovat. Takové letadlo, letící přes oceán z Londýna do New Yorku vygeneruje okolo 650 TB dat. Tedy ve větším měřítku, když spočítáme, že každý den v roce jen ve Spojených státech Amerických vzlétne okolo 28tisíc letadel, při průměrné době letu 6 hodin, tak vyprodukují tato letadla za rok asi 2,5 exabytu[1] surových dat. Takové množství však není možné v reálném čase zpracovat běžně dostupnými přístroji a tak se pouze ukládají, pro možné pozdější využití u případného problému, který se muže vyskytnout.

[1] exabyte odpovídá velikosti 10^{18} bytu

V těchto datech se mohou vyskytovat velice vzácné informace, například z dat tryskových motorů a senzorů snímající vnější prostředí lze předpovídat, které komponenty by se mohly v nejbližší době porouchat. Můžeme tedy zajistit jejich včasnou výměnu. Zamezí se tím způsobení škody, kterou by tato vadná komponenta napáchala, kdyby přestala řádně fungovat. A co nejvíce, záchranu lidských životů, které by se na palubě letadla v době havárie nacházely.

2.2 Odkud data pocházejí

Letecký průmysl ale není jediný velký producent dat. Když se koukneme do strojnictví, hutnictví, lékařství nebo i jiná odvětví, najdeme přístroje, které generují více dat, než která jsou schopna analyzovat. V dřívějších letech minulého století stroje také generovaly velké množství dat, ale jejich skladování bylo finančně nákladné a proto se nevyplatilo tuto data uchovávat. V současné době stojí běžné 3,5" pevné disky ve velikosti 2 TB něco přes 2000 Kč a disky o velikosti 3 TB se prodávají i pod hranicí 3000 Kč [3]. Pokud tedy cenu pevných disků přepočítáme podle toho, kolik zaplatíme za 1 GB dat, vyjde nám cena okolo 2 Kč. Ukládat data pro pozdější analyzování a jejich využití je tedy v současné době mnohem výhodnější než kdysi. Znázornění dat kolik jsme platili průměrně za 1 MB[2] dat, si můžete prohlédnout na obrázku (viz Obrázek 2 - Historický vývoj ceny za 1 MB dat).

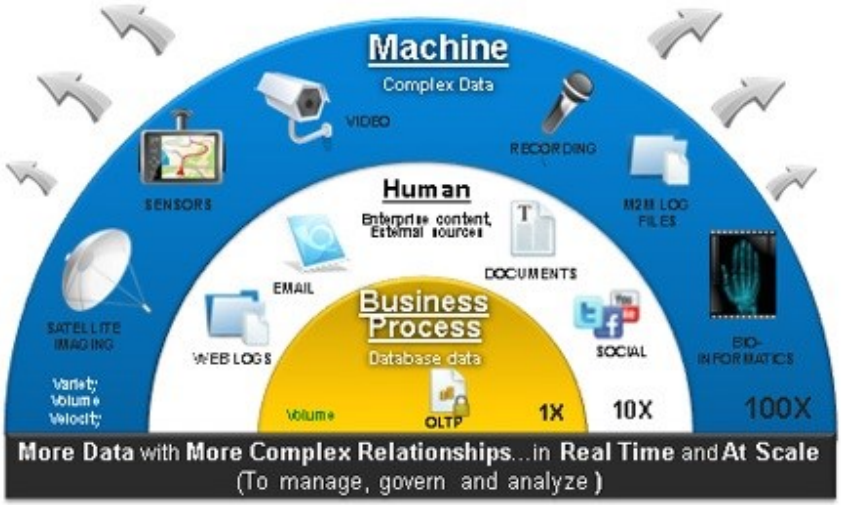


Obrázek 2 - Historický vývoj ceny za 1 MB dat [4]

Data jsou produkována v různých měřítkách a formách. Nejúspornější forma dat na náročnost úložiště nám představují data ukládaná do běžných relačních databázových modelů (viz Obrázek 3 - Zdroje a typy dat). V druhé kategorii se nachází data produkovaná lidmi používající běžné kancelářské balíky programů, jako je například webový prohlížeč, poštovní klient, programy pro zpracovávání textu nebo

[2] Bylo použito měřítko 10^6 bytu, tedy 1 MB, z důvodu jednoduššího srovnání cen v historii a současnosti. Teprve od roku 2000 cena za 1 MB natolik klesla, že se začala přepočítávat cena za 1 GB.

sociální média. V nejvyšší kategorii se pak nachází data nejobjemnější. Jsou převážně produkována stroji a jejich náročnost na úložiště je mnohdy velice vysoká. Taková data pochází například ze satelitního snímaní Země či vesmíru, tvorba audio a video nahrávek nebo také bioinformatika, která se zabývá shromažďováním, analýzou a vizualizací rozsáhlých biologických dat.



Obrázek 3 - Zdroje a typy dat [5]

2.3 Kam a co dělat s daty

Velikost dat ale není jediným problémem, se kterými se musí člověk potýkat. Data se ukládají v různých tvarech. Nejideálnější by bylo, kdyby se všechna data dala uložit ve strukturované podobě. To znamená, že se vejdou svým tvarem do klasické tabulky relační databáze, která je pro tento tvar dat optimalizována. To ale mnohdy není možné, a proto rozlišujeme dále data na strukturovaná, semistrukturovaná a nestrukturovaná. Jak názvy napovídají, jedná se o data, která mají určitou strukturu, a není problém je vložit do relačních databázových tabulek. Poté data, která mají částečnou strukturu, tedy semistrukturovaná. A nakonec data, která není vhodná strukturovat do relačních databází nebo to u nich není možné.

2.3.1 Semistrukturovaná data

Za semistrukturovaná data se nepovažují data, která nejsou surová, holá nebo data v konvenčních databázových systémech [6]. Semistrukturovaná data jsou tvořena pomocí danných pravidel, ale nejedná se o pravidla struktur běžného relačního modelu. Za struktury relačního modelu se obvykle považují tabulky nebo také grafy. Struktura semistrukturovaných dat nebývá jednoduše zjistitelná či viditelná, protože je tvořena podle určitého pravidla, které nám může být neznámo, a proto samotné prvky takové struktury je náročné rozdělit na samotná pole a přidat je do relační databáze. Obvykle se za semistrukturovaná data považují soubory formátu XML, EDI nebo také emaily.

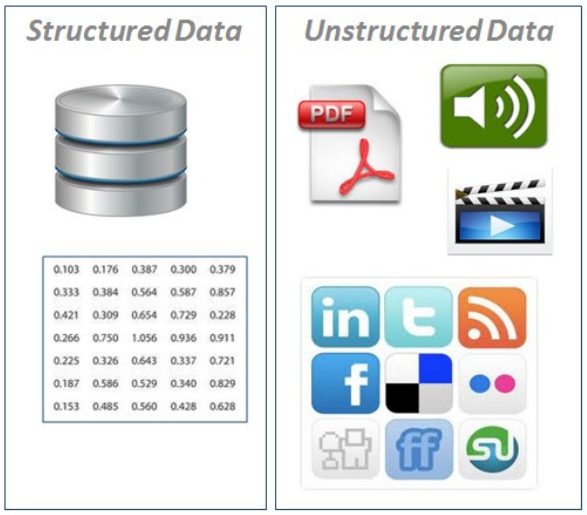
2.3.2 Nestrukturovaná data

Nestrukturovaná data jsou doplňkem pro strukturovaná a semistrukturovaná data. Samotná nestrukturovaná data, jak napovídá název, nemají vlastní danou strukturu, podle které bychom je mohli jednoduše roztřídit. Jedná se například o texty různých formátů, ať už to jsou elektronické knížky, dokumenty nebo prezentace [7]. Mezi nestrukturovaná data se ale také považují audio soubory, webové stránky, fotografie, filmy či videa. S takovými typy dat se velice obtížně pracuje a představují 80 až 90% dat,

které jsou v současné době ukládána. Navíc obsah těchto dat je mnohdy velice vzácný a obsahuje důležité informace, které by bylo vhodné zanalyzovat. Což vůbec není jednoduché, a proto se většinou pouze ukládají, pro budoucí zpětné zpracování.

2.3.3 Metadata

K nestrukturovaným souborům bývají obvykle přidávána tzv. metadata. Metadata, obecně nazývána jako data o datech, jsou záznamy, který nám upřesňují nebo popisují daný soubor. Například když vložíme na Facebooku nový status na zeď. Status uživatele obsahuje nejen text, který napíšeme do příslušného okénka nebo fotky, kterou ke statusu přiložíme, ale také metadata, která uživatel přímo nenapíše nebo nezadá, ať již vědomě či nevědomě. Jedná se například o čas, kdy jsme tento status odeslali nebo oblast, ve které se nacházíme nebo model zařízení, ze kterého jsme status odeslali. Jestli jsme status odesílali z mobilního klienta, kde se dále muže rozlišovat operační systém mobilního telefonu (Android, Windows Phone…) nebo přímo aplikace, která nám umožnila přístup k našemu Facebook účtu (Facebook aplikace, internetový prohlížeč…). Většinou uživatel nemá možnost měnit nebo tvořit tyto metadata, protože je dodatečně vytváří samotná aplikace. Proto medatada bývají strukturovaná a pomocí nich lze nestrukturovaná data jednodušeji kategorizovat.



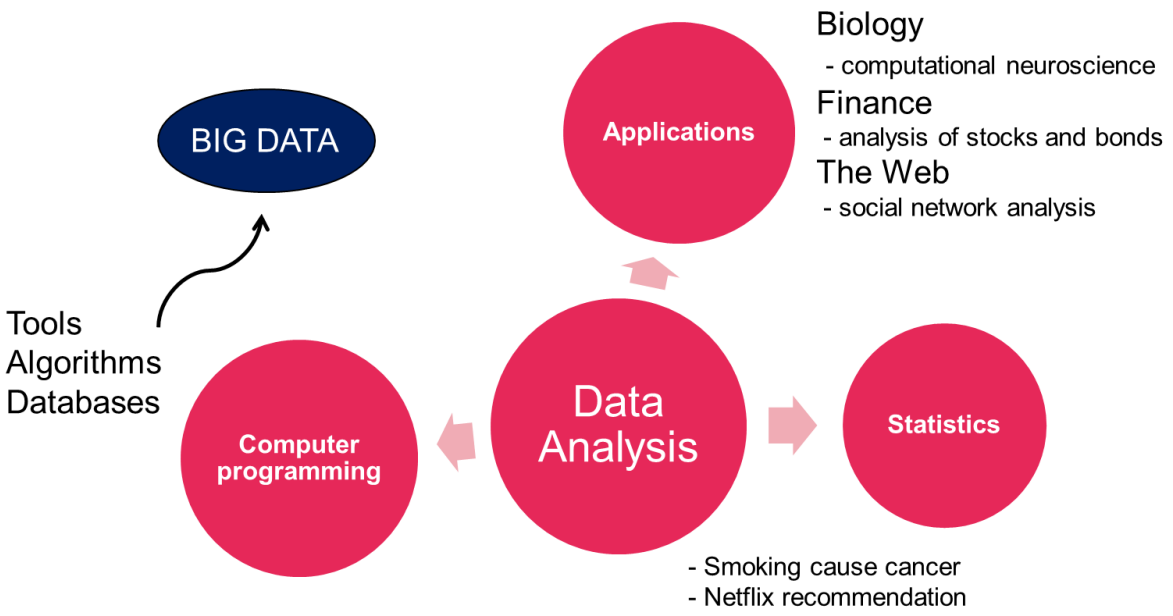
Obrázek 4 - strukturovaná a nestrukturovaná data [7]

V dnešní době může kdokoli svá data kamkoli ukládat v jakékoli podobě. Obrovská výhoda nestrukturovaných dat spočívá v jednoduchém uložení například na souborový disk. Běžný uživatel si udělá maximálně souborovou strukturu podle složek, ve kterých se pak orientuje. Až při ukládání souboru si program sám vytvoří pomocné metadata o tom, kdy byl soubor uložen, kým byl uložen nebo kdy byl změněn, popřípadě jakou verzí programu. Když si paní na personálním oddělení takto uloží za každý měsíc pro každého zaměstnance jeden výplatní lístek textového dokumentu, tak kdyby měla firma 1000 zaměstnanců, tak za jeden rok vytvoří až 12 000 dokumentů. Kdyby si chtěla spočítat, jaká byla průměrná mzda za jeden měsíc, tak musí projít 12 000 dokumentů a složitě si vypočítat, jaké hodnoty bude průměrná mzda dosahovat. Kdyby si ale tyto mzdy ukládala ve strukturované podobě, například do tabulkového editoru, kde každou hodnotu společnou pro všechny zaměstnance bude vkládat do stejného sloupečku, tak jednoduchým vzorcem na spočtení sumy jednoho sloupce k počtu zaměstnanců získá hodnotu průměrné mzdy, kterou potřebuje.

3 Datová analýza a Big Data

3.1 Datová analýza

Jedná se o proces, při kterém vyšetřujeme, čistíme či modelujeme data za účelem získání užitečných informací k budoucím účelům. Datová analýza je velice široký pojem, protože pro každý jiný typ dat nebo jiné odvětví, musíme vyvinout jiné principy, jak taková data analyzovat. Dokument ze soudního řízení se bude analyzovat jiným způsobem, než model auta, který zrovna technik navrhnul ve svém programu. Pro každé odvětví máme různé principy zpracování dat, tedy co je vhodné pro zpracování jednoho typu, může být naprosto nepoužitelné pro zpracovávání jiného typu, i když to tak zpočátku vypadá. Datová analýza většinou probíhá tak, že analytik vysloví nějakou hypotézu, kterou chce nad daty docílit. Dále musí vymyslet principy jak této hypotézy docílit a aplikovat tento princip na dotazovaná data, které jsou mu přístupná. Pokud analytik má vhodný princip a hypotézu, tak hypotézu potvrdí nebo vyvrátí. Například v lékařství můžeme aplikovat hypotézu výskytu rakoviny u lidí, kteří kouří. Pokud máme přístup k potřebným datům a aplikujeme na ně datovou analýzu, můžeme z nich vyvodit možný závěr, že kouření doopravdy způsobuje rakovinu. Taková data ale bývají v nestrukturovaných formách a velice obtížně se zpracovávají. Proto potřebujeme nástroje, které nám umožní tyto data analyzovat. A jedním z dostupných nástrojů je koncept Big Data.



Obrázek 5 - Datová analýza

3.2 Koncept Big Data

Velké a různorodé informace dnes bývají označovány jako Big Data (BD). Taková data však v překladu neznamenají pouze velké množství dat. Nejvýstižnější definice tohoto konceptu bývá definice pomoci 4V z anglického jazyka. Znamenají Volume (objem), Velocity (rychlost), Variety (různorodost), Veracity (věrohodnost).

3.2.1 Volume (objem)

Trend přibývání dat roste exponenciálně, a proto je to první z aspektu pro definování BD. Studie společnosti IDC Digital Universe [8] řekla, že už v roce 2011 přesáhla veškerá data za Zemi hranici 2 zettabytů[3], která neustále stoupá. Další studie firmy WIPRO [9] ukázala, že největší množství dat pochází ze Severní Ameriky. Také znázornili, že je celosvětově posláno asi 2,9 milionu emailů nebo za jeden den lidé na sociální síti Twitter napíší až 50 milionů tweetů (příspěvků). Podle WIPRO ale největší přírůstek dat zaznamenává celosvětově známý server YouTube. Na ten lidé nahrají každou minutu až 20 hodin videa.

3.2.2 Velocity (rychlost)

Abychom dokázali zpracovat zužitkovat data, musíme to umět dělat rychle. Rozlišujeme zde dva typy rychlostí. Jedna je schopnost zpracovat data v krátkém časovém měřítku, tzv. real-time processing. Jedná se o zpracování dat v reálném čase. Jak nám data přicházejí tak je okamžitě analyzujeme. Využíváno například na datech generovaná stroji v továrnách, kdy při chybě na páse potřebujeme okamžitě analyzovat záznamy a zjistit, že nastala někde chyba, aby se linky mohly okamžitě zastavit. Může se tak předejít velkým škodám při výrobě, díky níž by se jinak vyráběly vadné výrobky, než by se na danou vadu přišlo. Dalším aspektem je pak schopnost zpracovat velké množství záznamů v rozumném čase. Tedy, že nebudeme muset čekat na výsledek například celou noc, ale pouze pár minut.

3.2.3 Variety (různorodost)

Různorodá data se rozlišují na již zmíněná strukturovaná, semistrukturovaná a nestrukturovaná data. Klasické analytické nástroje nejsou schopné zpracovat různorodá data, a přitom taková data zastupují kolem 80% všech dat ve společnostech.

3.2.4 Veracity (věrohodnost)

Ve velkém množství dat se vyskytují i data, která nemusí být pravdivá, a aby zbytečně nezkreslovaly požadovaný výsledek, musí se k nim tak také přistupovat. Každý může na internet napsat příspěvek nebo článek, ze svého úhlu pohledu. A právě takový článek nemusí být nutně pravdivý nebo jeho uvěřitelná hodnota není příliš vysoká. Někdy se místo čtvrtého V (veracity, věrohodnost) objevuje element Value (hodnota), kde se zpracovává, nakolik může být zanalyzovaný výsledek užitečný.

[3] Velikost zettabytu odpovídá 10^{21}bytu

4 Apache Hadoop

Apache Hadoop je softwarová platforma, která se zabývá konceptem BD. Umožňuje práci s velkými daty, a jejich ukládání na běžně dostupný hardware. K jeho základnímu provozu tedy nepotřebujeme žádné speciálně upravené stroje, které by nám umožnily funkčnost této platformy. Dokáže propojit více stanic, se kterými pak pracuje, ale každá stanice zapojená do této platformy pracuje nezávisle na sobě. Vzniká nám tedy distribuovaný souborový systém Hadoopu (HDFS).

4.1 Vznik

Počátky distribuovaného souborového systému nalezneme v roce 2000. Firma Google začala vyvíjet svůj vlastní souborový systém GFS (Google File Systém), na trh se ale dostal až počátkem roku 2003. Nezávisle na GFS se o dva roky později pustila další firma do vlastní implementace souborového systém a to NDFS (Nutch Distributed File System). Založili ji tehdy dva vývojáři Dough Cutting a Mike Cafarelle. Tato firma položila základní kámen úspěchu pro budoucí Apache Hadoop. V roce 2005 přidali MapReduce do NDFS a vznikla Hadoop platforma od Apache, který byl pojmenován Doughem Cuttingem po hračce slona jeho syna. K prvnímu reálnému použití došlo až s další připojenou firmou Yahoo!, když v roce 2008 došlo ke zprovoznění clusteru, který byl sestaven z 10 000 stanic. Apache Hadoop je otevřená platforma a v současné době ji využívá mnoho dalších velkých internetových společností, jako:

- Yahoo!
- Microsoft
- Facebook
- Twitter
- Amazon
- eBay
- a spousta dalších [10]

4.2 Důvod přechodu z jiných platforem

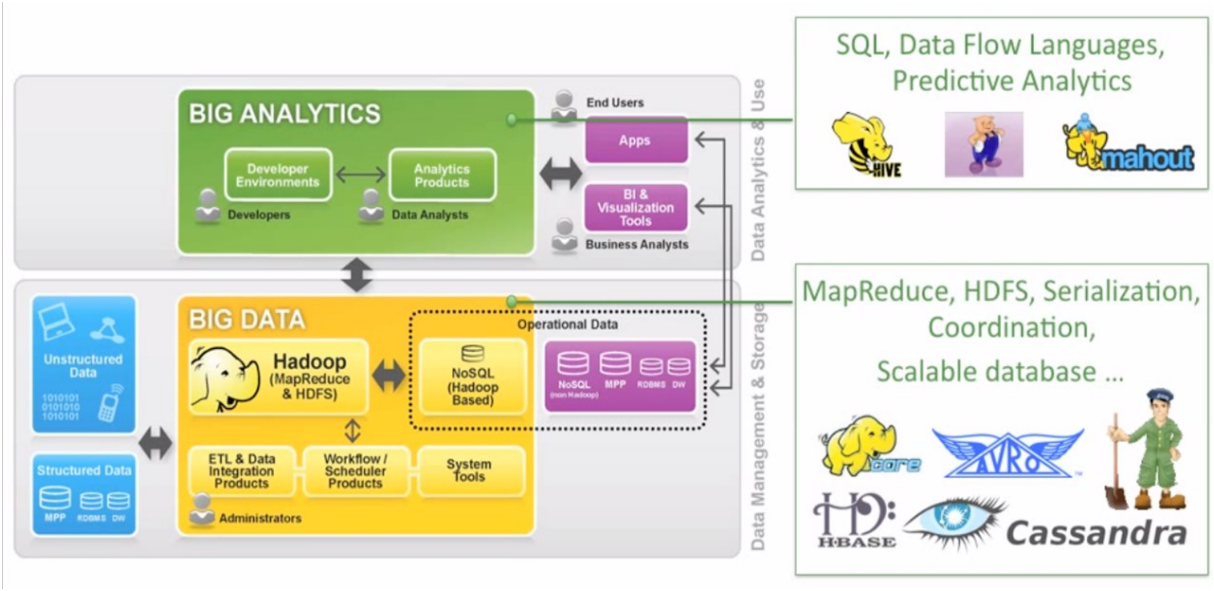
Důvod přechodu z klasických relačních databázových systémů je jednoduchý. Jeden z problémů klasické relační databáze je směr vývoje pevných disků. Přenosová rychlost na nebo z pevného disku postupuje mnohem rychlejším tempem než samotná vyhledávací doba na plotnách, kterými je pevný disk osazen. Tzv. seek time, znamená dobu, kterou se pevný disk připravuje ke čtení dat ze svých ploten. Hlavička pevného disku musí prvně nalézt správnou pozici na plotně a pak se plotny správně natočit do polohy, ze které se můžou začít číst data. Čím více přístupů a přesunů hlavičky je potřeba tím větší náročnost a zdržení ve čtených datech nastává.

V současné době lze tento problém řešit pomoci SSD (Solid State Disc), což je paměťové médium bez mechanických částí. Přístupovou dobu mají velice nízkou a čtecí rychlosti takového disku bývají velice vysoké i při náhodném přístupu k datům, které se na disku nacházejí. Tato alternativa však není levná a při porovnání ceny proti klasickým pevným diskům vychází toto řešení až 20násobně dráž. Cena při použití SSD za 1 GB se v současnosti pohybuje kolem 20 Kč [11]. Avšak ceny těchto SSD disků neustále klesají s jejich vývojem na trhu a za pár let může být jejich pozice úplně opačná.

Velká výhoda relačních databází je jejich jednoduchost implementace na menším objemu dat, kde data jsou ve strukturovaných formách. Ale čím více bude tato relační databáze narůstat v objemu dat, tím obtížněji se s ní bude pracovat, až dosáhne fáze, kdy již nebude úplně vhodná nebo přímo nepoužitelná jako primární zdroj úložiště dat. Takovou databázi pak lze například rozšířit nebo nakoupit silnější a kvalitnější vybaveni, které bude finančně velice náročné. Takové řešení bude ale stále jen dočasným řešením, protože se v budoucnosti bude potýkat se stejným problémem a řešení silnějšího nebo kvalitnějšího vybavení již nemusí být možné.

4.3 Architektura Hadoop

Architektura Hadoopu spočívá v zapojení běžně dostupných stanic do HDFS. Hlavní princip této architektury je zajistit rychlý přístup k datům, její spolehlivost a jednoduchá rozšiřitelnost pro budoucí potřeby. Samotný Hadoop je souborem různých programů, každá zaměřující se na jiné části použití.



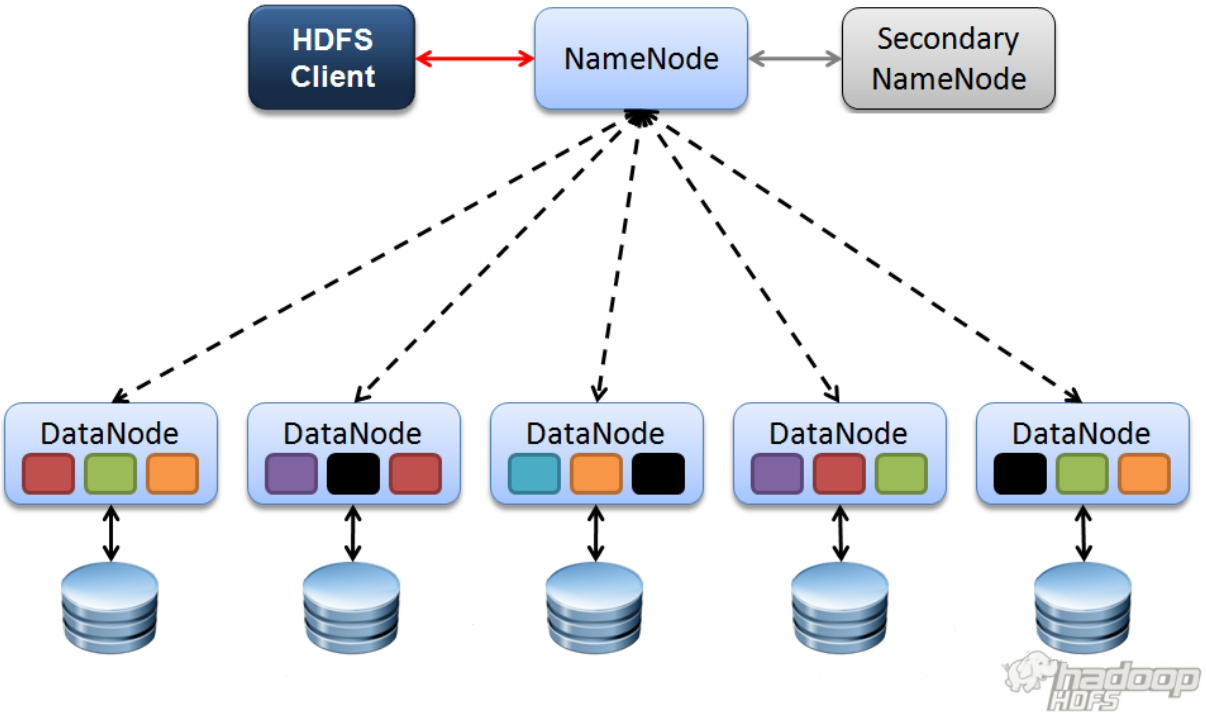
Obrázek 6 - Architektura Hadoop [12]

Celá architektura se muže rozdělit na dvě části. Na vrchní analytickou část a spodní datovou část (viz Obrázek 6 - Architektura Hadoop). Proces nám začíná načtením zdrojových strukturovaných nebo nestrukturovaných dat do Hadoop clusteru. Tento cluster se obecně skládá z většího počtu počítačů, které obsahují sdílený souborový systém. Tento proces je vhodný ke zpracování výpočetně náročných úloh. Největší omezení systému nastává v propustnosti linek při přístupu k velkému objemu dat, kvůli omezené datové propustnosti linky mezi jednotlivými uzly clusteru.

Analytická část, ve vrchní části obrázku je dostupná pro business analytiky nebo koncové uživatele. Ti různými způsoby přistupují k operačním datům. A jelikož se nejedná o klasické databázové sklady, nemohou použít běžné SQL jazyky pro dotazování nad daty, ale musí použít NoSQL jazyky aby získali potřebná data. Zde se nabízí například open source Hive pro Hadoop [12]. Hive slouží jako překladač HiveSQL dotazů na NoSQL dotazovací jazyk, který je psán v java kódů. Tento jazyk umožní dotazování nad uloženými daty v operační paměti v Hadoop clusteru. Nad těmito daty pak lze v analytické části vytvářet různé reporty, statistiky, určené pro koncové uživatele (například manažery).

4.4 Hadoop cluster

„Hadoop cluster je speciální typ výpočetního clusteru navrženého speciálně pro ukládání a analyzování velkých objemů nestrukturovaných dat v distribuovaném výpočetním prostředí“ [13]. Tento systém se skládá z velkého počtu běžných stanic, na které se autoritativně ukládají data, která potřebujeme uskladnit. Každá stanice, DataNode, má svůj vlastní výpočetní výkon a své úložiště. Stanice mezi sebou nespolupracují k dosažení cíleného výsledku. Každá má za úkol zpracovat svou přidělenou úlohu, kterou ji zaslal její nadřízený. Tomuto nadřízenému prvku se říká NameNode (viz Obrázek 7 - Architektura HDFS).



Obrázek 7 - Architektura HDFS [14]

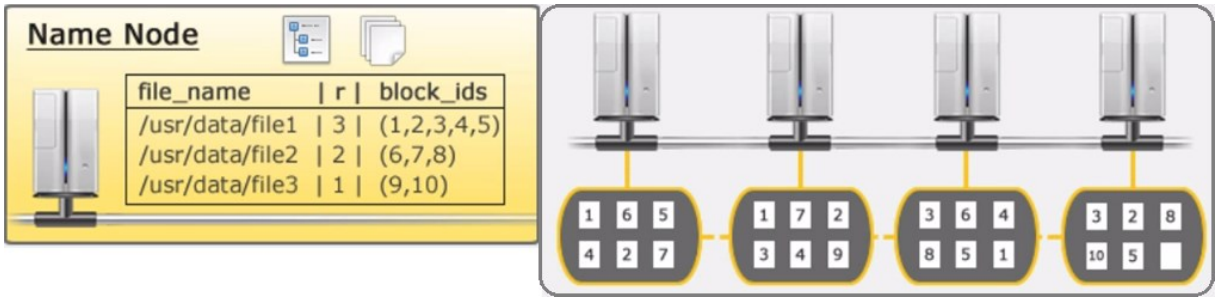
4.4.1 NameNode

Nejdůležitější prvek v Hadoop clusteru, který nám vytváří distribuovaný souborový systém, také známý jako Hadoop Distributed File Systém (HDFS). Umožňuje tvorbu distribuovaných výpočtů v HDFS na principu master-slave. Řídí nám tok dat mezi DataNody a uchovává si metadata o souborovém systému, který je k NameNodu připojen. Tyto metadata obsahují informace o tom, kde se jednotlivá data uložila, popřípadě jejich kopie na jiném DataNodu. Také musí znát aktuální strukturu HDFS, takže neustále NameNode testuje, zda každý DataNode je aktivní. NameNode si neustále žádá odezvu od DataNodů a pokud se přestane ozývat nějaký DataNode, označí si jej za neaktivní. NameNode ve svých metadatech má uvedeno, která data se na tomto neaktivním DataNodu nacházela a tyto data okamžitě replikuje na jiný DataNode aby o ně nepřišel. NameNode se neustále snaží udržovat více kopií jednoho souboru v HDFS, hlavně kvůli výpadku nějakého DataNodu. Nepřijde tudíž o data, která se na odstaveném DataNodu nacházela. Tomuto procesu se říká Replikační faktor (RF).

NameNode se neustále snaží rozprostírat data uvnitř HDFS mezi jednotlivé DataNody tak, aby byly pokud možno co nejrovnoměrněji zaplněné. Pokud se tedy připojí nový prázdný DataNode do HDFS a NameNode si jej označí za aktivní, okamžitě rozprostře data z ostatních DataNodu tak, aby opět byly všechny rovnoměrně zaplněné. Umožní tím rozprostření náročnosti jednotlivých úloh mezi vícero DataNodů. Tento proces je sice náročnější, ale plně automatický ze strany NameNodu. Tím nám vzniká úložiště, které je velice jednoduše rozšiřitelné o další stroje pro zvětšení skladovací i výpočetní kapacity HDFS.

4.4.1.1 *Replikační faktor*

Základní princip při replikaci dat je takový, aby se jeden soubor nenacházel na té samé stanici jako původní soubor, ze kterého kopie vzniká (viz Obrázek 8 - Replikační faktor v DataNodech). Důvod je jednoduchý. Pokud přijdeme o stanici, kde se nachází zdrojový i replikovaný soubor, data budou nenávratně pryč. Základním standardem bývá nastavení RF na 3. Při nastaveném RF 3 se nachází jeden soubor na třech různých DataNodech v HDFS, tedy jeden původní zdrojový soubor a jeho dvě kopie.



Obrázek 8 - Replikační faktor v DataNodech [15]

RF lze měnit podle požadavků, které jsou na systém kladeny. Čím nižší je nastavený RF, tím klade menší požadavky na úložný prostor pro data. Pokud máme dostupných 1000 TB úložného prostoru, tak při použití RF 4 se nám zmenší reálná kapacita úložiště na 250 TB (tedy 1000/4). Při použití toho samého úložiště a pouhým navýšením RF na 10, reálná kapacita úložiště již klesne 10x oproti původní kapacitě a 2,5krát oproti původnímu RF na hodnotu 100 TB (tedy 1000/10). Vyšší RF nám ale přináší značná pozitiva. Jedním je vyšší bezpečnost uchovávaných dat. Když se soubor nachází na více místech, tím méně pravděpodobnější je, že i po výpadku více DataNodů nám soubor stále zůstane zachován. Druhým pozitivem je, že při zpracování požadavku řeší problém více stanic. Tedy data jsou přijímána od té stanice, která jako první splní požadovaný příkaz nebo přijmutí výsledků z více stanic pro kontrolu, že výsledky byly správně zpracovány.

4.4.2 Secondary NameNode

Nazýván také jako BackupNode. Pracuje jako záložní stanice NameNode. Procesy, které vykonává primární NameNode se zaznamenávají na zálohu, z důvodu výpadku primárního by přestal kompletně pracovat celý HDFS. Rozdíl oproti NameNode je, že nezískává informace ze změn v souborovém systému v reálném čase. Záloha metadat se tvoří v pravidelných časových intervalech. Nezajistí nám dokonalou náhradu primárního NameNode v případě jeho výpadku, pouze nám minimalizuje výpadek HDFS po kterou bude nefunkční a ztrátu dat, která byla provedena od posledního intervalu zálohy. Za

primární i sekundární NameNode by se měly vybírat kvalitní stanice, které nám sníží možné riziko výpadku a ztráty dat.

4.4.3 DataNode

DataNode slouží jako jednotka pro ukládání a práci s daty v HDFS. Každý DataNode má vlastní blok s úložištěm a vlastní výpočetní výkon pro práci nad daty ve vlastním bloku. NameNode řídí ukládání a práci s daty, kterou mají jednotlivé DataNody zpracovat na svém bloku. DataNody spolu vzájemně komunikují v případě replikování dat, aby byl dodržen RF. Tento proces sice stále řídí NameNode, ale replikovaná data si mezi sebou přeposílají jednotlivé DataNody. NameNodu po ukončení procesu zašlou zprávu ohledně úspěšné/neúspěšné replikaci dat.

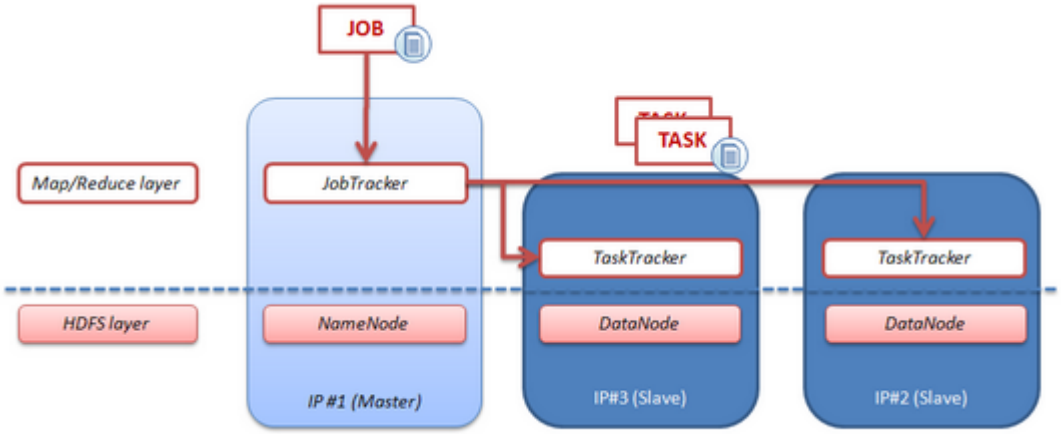
DataNode musí neustále oznamovat NameNodu svoji aktivitu, tzv. heartbeat. Tím DataNode neustále oznamuje, že nevykazuje žádný problém a je v bezproblémovém provozu. Tato zpráva je zasílána co tři sekundy pomocí TCP protokolu, tzv. handshake [16]. V každé desáté zprávě pak DataNode oznámí stav vlastního bloku, aby NameNode mohl aktualizovat svá metadata a dle potřeby zajistit dodržení RF.

4.4.4 JobTracker

JobTracker je služba Hadoopu, která zajišťuje přímé spojení mezi aplikací uživatele a Hadoop clusterem. Tato služba ovládá úlohu MapReduce a má nad ní plnou kontrolu. Přiřazuje úlohy jednotlivým uzlům a monitoruje jejich současný stav. Pokud nastane na nějakém uzlu selhání při řešeném úkolu, restartuje řešený průběh a zašle požadavek jinému uzlu. Zpravidla je JobTracker nasazován na stejnou stanici jako NameNode, který přímo zadává a kontroluje stavy jednotlivých uzlů. JobTracker má tedy neustále aktuální informace ohledně HDFS.

4.4.4.1 Průběh JobTrackeru: [17]

1. Uživatelská aplikace zašle požadavek do JobTrackeru.
2. JobTracker požádá NameNode k určení pozic požadovaných dat.
3. JobTracker zjistí pozice TaskTracker uzlů s dostupnými pozicemi na nebo v blízkosti dat.
4. JobTracker zašle vybraným uzlům TaskTrackeru práci k vykonání.
5. Uzly TaskTrackeru jsou neustále sledovány a zaznamenáván jejich průběh. Pokud uzel nezasílá své heartbeat v pravidelných intervalech, považuje se za neaktivní a požadovaná práce je přirazena jinému TaskTrackeru.
6. Pokud selže nějaká operace na TaskTrackeru, oznámí tuto chybu JobTrackeru. JobTracker poté rozhodne, jak bude postupovat. Zadaný úkol může přeposlat na jiný uzel, zaznamenat vyhnutí se určitému záznamu nebo může vyřadit TaskTracker jako neaktivní uzel, kvůli nespolehlivosti.
7. Pokud je zadaná práce TaskTrackeru celá hotová, JobTracker aktualizuje svou dostupnost.
8. Klientská aplikace může získat informace z JobTrackeru.



Obrázek 9 - Průběh úkolů JobTrackeru a TaskTrackeru [18]

4.4.5 TaskTracker

TaskTracker je zodpovědný za provedení zadané úlohy od JobTrackeru na svém uzlu s daty. Pracuje na principu master-slave architektury. Za řídící prvek se považuje JobTracker, který přerozděluje různým TaskTrackerům jednotlivé úlohy. Každý TaskTracker musí při zadaném úkolu odesílat v pravidelných intervalech heartbeat, ve kterém oznamuje svou funkčnost a stav v řešení zadaného úkolu. Pokud TaskTracker přestane odesílat heartbeat, je označen za neaktivní a JobTracker přesune zadanou úlohu na jiný, aktivní, TaskTracker. Každý TaskTracker řídí procesy na svém datovém uzlu a přiřazuje jednotlivé individuální úlohy podřízeným stanicím.

4.5 Paralelní zpracování dat v HDFS

Paralelní zpracování dat lze rozdělit podle dvou architektur. Paralelní zpracování na sdílené paměti a paralelní zpracování na distribuované paměti.

4.5.1 Systémy se sdílenou pamětí

U paralelního zpracování na sdílené paměti komunikují procesory prostřednictvím jedné paměti, ke které mají společný přístup. Jednotlivé procesory tak mohou být specializovány na určitý úkol. Nevýhoda tohoto systému je, že procesory musí synchronizovat své aktivity na společné lince, protože v jednu chvíli muže do paměti vstupovat pouze jeden procesor, řeší se tedy problém exkluzivního přístupu do paměti.

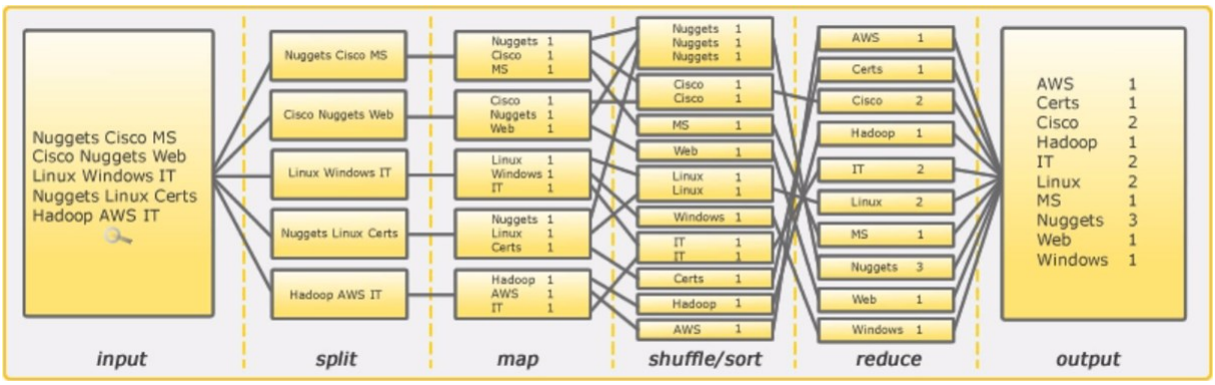
4.5.2 Systémy s distribuovanou pamětí

Jedná se zpravidla o velké množství strojů, které jsou mezi sebou spojeny komunikační linkou. Každý procesor pracuje s vlastní pamětí, tedy odpadá problém řešení exkluzivního přístupu do paměti. Řešit se ale musí problém v komunikaci mezi jednotlivými procesory. A jak již název napovídá tak tento typ systému s distribuovanou pamětí je využíván v HDFS.

4.5.3 MapReduce

Pro způsob paralelního zpracování dat s distribuovanou pamětí byla společností Google v roce 2004 vyvinuta softwarová platforma MapReduce (M/R) [19]. M/R platforma dovoluje vývojářům vyvinout programy, které jsou schopny zpracovávat velké objemy dat na paralelních systémech tvořených systémy s distribuovanou pamětí.

M/R rozděluje svůj průběh na 6 samostatných procesů (viz Obrázek 10 - Proces MapReduce). Pro zjednodušení se ale uvádí pouze proces map a reduce. Platforma M/R se skládá z instance JobTracker a TaskTracker. Jednotlivé instance JobTracker řídí zadáváním jednotlivých úloh TaskTrackerům, které jsou zodpovědné za zpracování této úlohy na svém uzlu. Jak již bylo uvedeno v předchozí kapitole, TaskTracker neustále oznamuje svůj aktuální stav pomocí hearthbeat, a proto JobTracker ví v jaké fázi celého procesu M/R se nachází.



Obrázek 10 - Proces MapReduce [20]

4.5.3.1 Input

Proces začíná požadavkem od JobTrackeru lokalizovat stanice, na kterých se nachází data, se kterými potřebujeme pracovat. Jednotlivým stanicím, na kterých je spuštěn TaskTracker, je zaslán požadavek na práci s daty. Pokud nějaká stanice není schopná přistoupit k datům, požadavek se přesune na jinou stanici.

4.5.3.2 Split

Tato funkce obstará rozdělení celků na jednotlivé části, které si můžeme definovat podle obsahu. Z obrázku (viz Obrázek 10 - Proces MapReduce) jde vidět, že vstupní data funkce split rozdělila podle řádků.

4.5.3.3 Map

Funkce Map rozdělí rozdělené části z předchozího procesu split na co nejmenší části. V případě textu na slova. Rozdělovač může být různě definován, kterým určíme pravidla rozdělování (mezera, tečka, čárka, středník, atd.). Dále určí jednotlivé klíče pro rozdělené prvky a počet výskytů (v této fázi je počet výskytů pro všechny prvky zvlášť rovno jedné).

4.5.3.4 Shuffle/sort

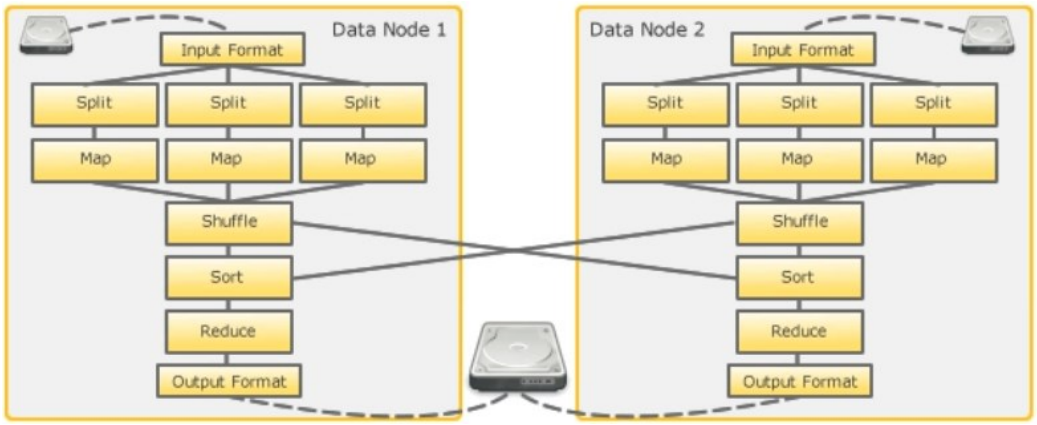
Jednotlivé části, rozdělené ve funkci Map, jsou přeuspořádány podle klíčových prvků do listu. Každý list obsahuje klíčový prvek, hodnotu prvku a list s počtem výskytů jednotlivých klíčových prvků. Jednotlivé listy před posláním do funkce Reduce se seřadí podle zadaného kritéria.

4.5.3.5 Reduce

Reduce část má na starost agregaci dat. Na svůj vstup dostane seřazené listy, ve kterých agreguje počet výskytů jednotlivých prvků. Pro každý prvek tedy spočítá hodnotu, kolikrát se ve zdrojovém souboru nacházel.

4.5.3.6 Output

Závěrečná část získá již agregovaná data. Output zajistí výstup dat v potřebném formátu a zašle hotová data JobTrackeru. Pokud celá úloha byla úspěšně ukončena, tak je v JobTrackeru zařazena mezi hotové úlohy a výstup je zobrazen uživateli.



Obrázek 11 - Paralelizace MapReduce [20]

4.5.4 Použití N-gramů

Obecně se n-gram dá vyjádřit jako sled *n* po sobě jdoucích položek z určité posloupnosti. Konkrétněji v případě zpracovávání textu se n-gram bude zabývat sledem *n* po sobě jdoucích slov. Sled dvou po sobě jdoucích slov je označován s předponou bi-, bigram. Pro vyšší sled po sobě jdoucí slov se použije náležitá předpona z latinského jazyka. Pro tři tri-, čtyři quadra- a tak dále.

N-gramy jsou velice oblíbeným nástrojem pro zpracovávání nestrukturovaného textu. Pomocí vytvořených pravidel se relativně snadno dá zjistit, jaké n-gramy se v textu nejčastěji objevují, a podle toho se může text kategorizovat nebo zjistit jinak skrytou informaci v rozsáhlém textu. Je ale nutno dodržet určitá pravidla pro tvorbu n-gramů. Ne všechny slovní posloupnosti jsou vhodné k tomu, aby se považovaly za vhodný n-gram. Ve výsledku n-gramů by se neměly vyskytovat předložky, spojky a zájmena, protože jejich využitelná hodnota je mnohdy nežádoucí, v anglickém jazyce je pro tyto slova vyhrazen termín „stopwords“. Z tohoto důvodu je tedy důležité vědět v jakém jazyce je psán analyzovaný text, neboť každý jazyk obsahuje různá nežádoucí slova.

4.5.4.1 Možné využití n-gramů [21]

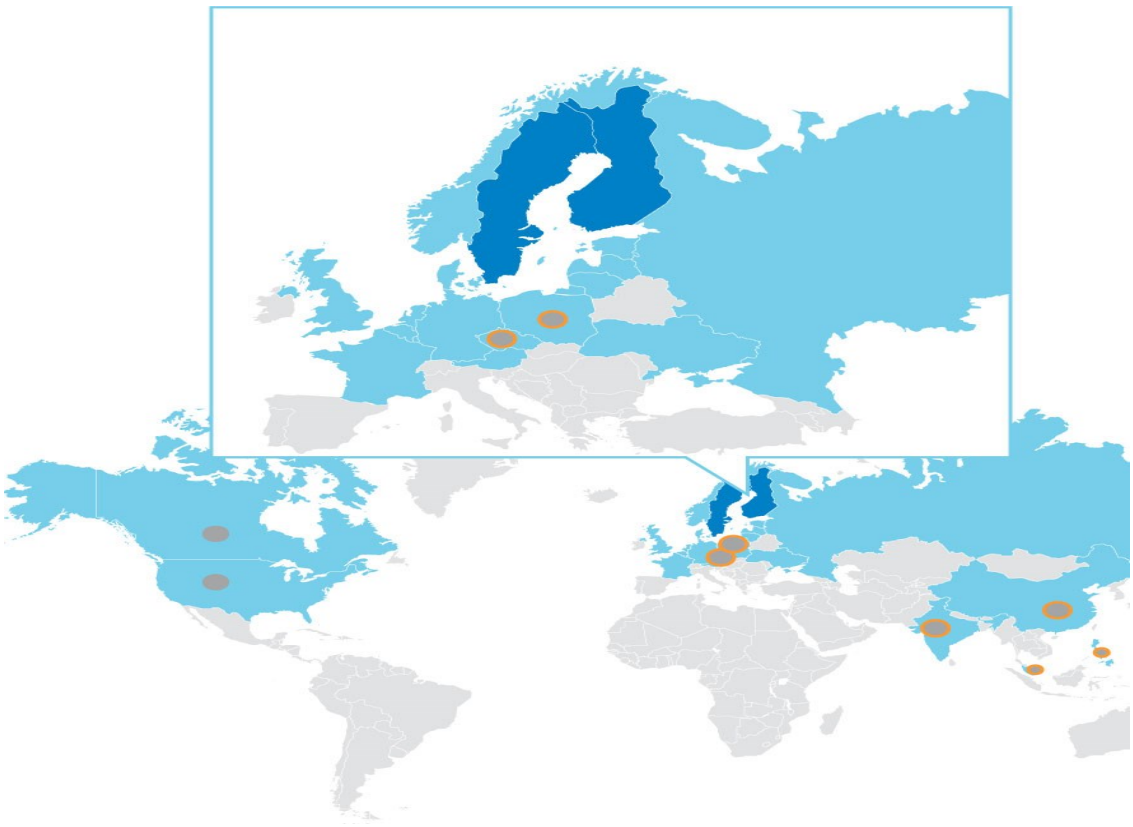
- (n-gramy) Vyhledání důležitých témat v textu v závislosti na listu předem vybraných slov.
- (n-gramy) Vyhledání oblíbených témat bez specifických požadavků na vstupu.
- (kontextové n-gramy) Vyhledávání souvislostí pro určitá slova (např.: „Twitter je ___“).
- (n-gramy) Vyhledání často používaných URL sekvencí.
- (kontextové n-gramy) Vyhledání často používaných URL sekvencí, které začínají nebo končí určitou částí URL.
- (kontextové n-gramy) Předpovídání běžně používaných slov při vyhledávání.

5 Využité systémy a zdroje dat

V následující kapitole Vás seznámím se zadáním projektu, jehož část byla realizována v projektu jako praktický výstup z bakalářské práce. Jelikož se pracuje s daty skutečných zákazníků, kteří jsou do tohoto projektu zapojeni, tak citlivá data o těchto zákaznících nesmí být uveřejněna. Konečné analýze a výstupu dat v prezentační formě se budu věnovat v následujících kapitolách.

5.1 Dodavatelská společnost

Projekt, který je v této práci řešen, byl zadán společnosti Tieto za účelem získání kompetencí pro práci s BD a jejich přínos pro zadavatelskou firmu. Tieto je současné době největší severoevropský dodavatel IT služeb, který poskytuje komplexní nabídku služeb v IT oblasti, jak pro soukromý, tak i veřejný sektor. Síť pokrytí společnosti Tieto a zobrazení největších poboček společnosti je vyobrazeno na následujícím obrázku (viz Obrázek 12 – Mapa působení Tieta).



Obrázek 12 – Mapa působení Tieta [22]

Počet zaměstnanců Tieta byl k 31. Prosinci 2013 celkem 14 699, kteří se rozprostírají po různých zemí světa [22]. Počet zemí, ve kterých Tieto operuje je okolo 20. Nejvíce zaměstnanců pracuje ve Finsku a Švédsku. Hned za těmito zeměmi se z hlediska počtu zaměstnanců rozprostírá česká pobočka Tieta. V České republice pracuje okolo 2 000 zaměstnanců, z nichž většina pracuje v Ostravských pracovištích Tieta. V Ostravě tak vznikla nová moderní budova stavěná speciálně pro potřeby Tieta, pojmenovaná Ostrava Tieto Towers, která umožnila přesunutí zaměstnanců z více poboček v Ostravě do jednoho místa.

5.2 Zadání projektu

V rámci práce na projektu pro zadavatelskou společnost, byla nabídnuta realizace řešení pomocí konceptu Big Data a její možné přínosy pro zákazníka. Jednalo se tedy o domluvu, kdy zákazník v rámci dohodnutého projektu poskytne svá data, pro možné analyzování a zpracování, ze kterých získá cenné informace o své firmě, které nejsou součástí běžného BI řešení nabízených služeb.

Výstup z poskytnutých dat bude poskytnut v reportingovém programu QlikView, který zajistí a zobrazí výstup zpracovaných dat. Zákazník se poté rozhodně zda nasazení tohoto řešení má pro něj význam a se mu vyplatí využívání této platformy společně s poplatky za užívání licencí.

5.3 Zdrojová data

O systémech, ze kterých data pocházejí, nebyly poskytnuty žádné informace. Jedná se o logovací soubory a eventy, které jsou zachycovány na podpoře servisních služeb. Pocházejí z různých zdrojů a obsahují informace např. o výpadcích serverů, problémech stanic nebo požadavky pracovníků o vyřešení určitého problému. Tato data jsou generována z velké části samotnými strojovými systémy, ale ojediněle se zde vyskytují i požadavky zadané přímo pracovníky.

Zdrojová data byla poskytnuta od zákazníka v souborovém typu Flat file, konkrétněji v datovém souboru CSV. Ve specifikaci požadavků byla data textově doplněna o základní popisy a vysvětlivky.

Záznamy nejsou plně strukturované, protože pochází z více zdrojů a nejsou generována jedním systémem. Pro všechny záznamy ale platí určitá pravidla jednotlivých atributů, podle kterých je lze třídit a kategorizovat. Záznamy ze všech zdrojů musí hlavně obsahovat čas, kdy byly vygenerovány, čas kdy byly přijaty na podporu služeb pro vyřešení problému, pokud je problém vyřešen, tak čas vyřešení problému, text popisující uvedený problém a kým byl záznam generován (člověk, server, služba, program…).

5.4 Použité systémy

Praktickou část práce jsem vykonával na osobním počítači ve firmě Tieto. Na osobním počítači byl nainstalován operační systém Windows 8.1 Pro, a základní specifikace počítače byly procesor Intel Core i5-2430M 2,4GHz, 8GB operační paměti a pro úložiště dat SSD disk Crucial M500. Hadoop cluster společně s Hivem byl nakonfigurován ve virtuálním prostředí VirtualBox od společnosti Cloudera. Tento virtuální systém je nakonfigurován jako single node Hadoop cluster. Výpočetní výkon tohoto řešení je tedy omezen lokálním strojem, na kterém běží a jaké prostředky se tomuto virtuálnímu stroji vyčlení.

Hadoop byl nainstalován ve verzi 2.5.0 a Hive, který vykonával funkci MapReduce ve verzi 0.13.1, plně kompatibilní s použitou verzí Hadoopu.

Pro účely reportingu byl využit program QlikView, verze 11.20.11718.0 SR1, který byl schopen pomocí ODBC Data Source Administrator konektoru, verze 6.1.7601.17632, se napojit přímo na Hadoop cluster, kde se nacházela potřebná data.

5.5 Struktura vstupních dat

Data, která byla poskytnuta ke zpracovávání, pochází z interní databáze centra podpory služeb. Hlavní důvod, proč zpracovávat taková data pomocí platformy Hadoop je rozdílná struktura a absence dimenzionálních tabulek ve zdrojových datech. I když každý záznam obsahuje jistou strukturu, tak samotný problém a popis takového záznamu je vyjádřen v jediném atributu. Problémy, které se tedy na určité stanici nebo serveru vyskytnou, jsou popsány v textové části daného každého záznamu, což značně urychluje proces tvorby a jednoduchost oznámení incidentu. Vícero záznamů, ať již stejných nebo podobných, lze velice obtížně zpětně filtrovat podle zadaných kritérií, protože veškerá informativní hodnota incidentu je popsána v jediném atributu, tedy nestrukturovaném textu.

Soubor dostupných zdrojových dat je popsán v následující tabulce (viz Tabulka 1 - Zdrojové soubory).

Zdroj	počet atributů	počet záznamů	velikost
/user/omt/DIM_OMT_GROUPS.csv	5	2492	238 KB
/user/omt/FACT_AM_SUPPORT_SUMMARY.csv	46	1262826	589.7 MB
/user/tone/TONE_CHANGES_V.csv	19	3562	947 KB
/user/tone/TONE_INCIDENTS_V.csv	19	473655	129.7 MB
/user/tone/TONE_PROBLEMS_V.csv	15	574	138 KB
/user/events/BigData_Dropped_20131003.csv	19	4128883	1.0 GB
/user/events/events_201306.csv	15	26016894	6.3 GB
/user/events/events_201309_1_edit.csv	19	3876518	969.6 MB

Tabulka 1 - Zdrojové soubory

Tato data pochází přímo ze servisních služeb od zákazníka a obsahují velké množství interních dat několika společností. Pro účely mé práce budu využívat záznamy, týkající se různých problémů, jak na osobních stanicích pracovníků, tak automaticky generované záznamy serverů apod. Objemy zpracovávaných dat sice nespadají pod běžné měřítko BD, kde se zpracovávají mnohem větší kvanta dat. Avšak nestrukturovaná forma dat i v takovém malém množství může znamenat velký problém v následném zpracování.

Data vybraná pro zpracování pochází ze zdroje FACT_AM_SUPPORT_SUMMARY.csv, který obsahuje celkově 46 atributů. Ne všechny atributy jsou ale informativně zajímavé pro mé zpracování a tak je nutno provést analýzu, které atributy jsou nutné, vhodné nebo nevhodné. Pro jednodušší přehled nad zpracovávanými daty jsem také použil omezení dat pro 10 zákazníků, kteří mají vygenerováno nejvíce záznamů ve zdroji. Těchto 10 největších zákazníků zaujímá přes 95% všech dat ze zdroje, a tedy zpracování zbytku menších firem by mělo malou informativní hodnotu z důvodu malého množství dat, která by se pro takového zákazníka zpracovávala.

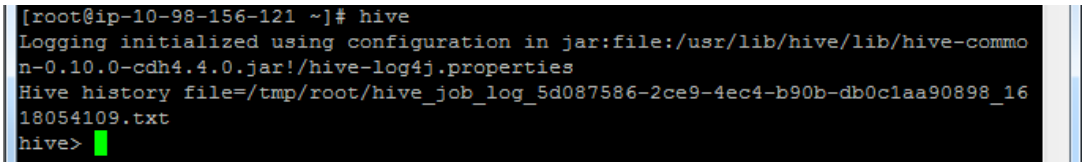
5.6 Vkládání dat do HDFS

Pro přístup do Hadoop clusteru je využíván terminál v OS Linux. S dostatečnými uživatelskými právy tak můžeme naimportovat zdrojová data do Hadoop clusteru pomocí příkazu *hadoop fs -put localfile /path/hadoopfile* [17]. Příkaz je nutné provést pro každou strukturu, kterou chceme importovat do Hadoop clusteru. V mém případě tedy potřebuji nahrát pouze strukturu FACT_AM_SUPPORT_SUMMARY.csv. Příkaz pro nahrání struktury bude vypadat následovně:

```
hadoop fs -put FACT_AM_SUPPORT_SUMMARY.csv /user/omt/FACT_AM_SUPPORT_SUMMARY.csv
```

5.7 Tvorba struktur v HIVE

Pokud je na serveru správně nakonfigurován Hive, spustí se po zadání příkazu *hive* do okna terminálu. Hive zde pracuje jako překladač z HiveQL, tedy jazyka, který je velice podobný rozšířenému SQL na Java jazyk, který je zpracováván Hadoop clusterem. Uživatel tedy není nucen znát složitou syntaxi Java kódu, kterou by se potřeboval naučit pro práci se strukturami v Hadoop clusteru.



Obrázek 13 - Spuštění překladače Hive v Hadoop clusteru

Nejprve se musí vytvořit tabulky pro nahrané struktury ze zdroje a poté jim přiřadit cestu k nahrané struktuře. Pro již nahranou strukturu FACT_AM_SUPPORT_SUMMARY.csv bude muset být vytvořena tabulka s příslušnými formáty pro 46 atributů. Typy formátů pro jednotlivé atributy byly popsány v dokumentaci ke zdrojovým datům, ze kterých mohla být vytvořena tabulka se strukturou uvedenou v Tabulka 2. Při vytváření tabulky je ještě nutno zadat cestu k nahrané struktuře, nacházející se v Hadoop clusteru, aby se rovnou naplnila daty. Nahrávat data do vytvořených tabulek lze udělat pomocí 2 způsobů. Zadáním cesty ke zdrojové struktuře nebo použití jiné již naplněné tabulky. Neexistuje zde běžně známý příkaz z SQL jazyka *INSERT INTO table*. V tomto případě chceme nahrát data ze souboru, takže se použije příkaz *LOAD.* Příkaz má následující strukturu.

```
LOAD data inpath 'path to source' into table destination_table
```

```
CREATE EXTERNAL TABLE FACT_AM_SUPPORT_SUMMARY
(
INCIDENT_ID                STRING,    SLA_COLOR                 STRING,
INCIDENT_TYPE              STRING,    LAST_1ST_TIER_TEAM        STRING,
INCIDENT_SUB_TYPE          STRING,    LAST_2ND_TIER_TEAM        STRING,
CONTRACT                   STRING,    LAST_3RD_TIER_TEAM        STRING,
SERVICE                    STRING,    SOLVED_YEAR_PERIOD        BIGINT,
SERVICE_CONTENT            STRING,    CREATED_YEAR_PERIOD       BIGINT,
SERVICE_DETAIL             STRING,    CREATE_TIME               TIMESTAMP,
IDENTIFIED_ITEM            STRING,    MUST_BE_SOLVED            TIMESTAMP,
```

COST_CENTER_ID	STRING,	WASSOLVED	TIMESTAMP,
RCVR_COST_CENTER_ID	STRING,	WASREACTEDTIMESTAMP	TIMESTAMP,
LEVEL_CODE	STRING,	MUST_BE_REACTED	TIMESTAMP,
RCVR_LEVEL_CODE	STRING,	SECOND_TIER_SOLVED_TIME-STAMP	TIMESTAMP,
ASSIGNED_TO_GROUP	STRING,	THIRD_TIER_SOLVED_TIMESTAMP	TIMESTAMP,
ASSIGNED_TO_INDIVIDUAL	STRING,	INCIDENT_DESCRIPTION	STRING,
SOLVED_BY_GROUP	STRING,	SOLUTION_VERIFICATION_NEEDED	INT,
SOLVED_BY_INDIVIDUAL	STRING,	INCIDENT_QTY	BIGINT,
CUSTOMER_ID	STRING,	REACTION_VIOLATIONS_QTY	BIGINT,
EMAIL	STRING,	SOLVING_VIOLATIONS_QTY	BIGINT,
CUSTOMER_NAME	STRING,	FIRST_SOLVING_VIOLATIONS_QTY	BIGINT,
CUSTOMER_ORGANIZATION_NAME	STRING,	SECOND_SOLVING_VIOLATI-ONS_QTY	BIGINT,
PRIORITY	INT,	PARTITION_KEY	BIGINT,
SOURCE	STRING,	GROUP_CATEGORY	STRING,
STATUS	STRING,	GROUP_NAME	STRING
)			
ROW FORMAT DELIMITED			
FIELDS TERMINATED BY "§"			
LOCATION '/user/omt/FACT_AM_SUPPORT_SUMMARY'			

Tabulka 2 - Struktura tabulky FACT_AM_SUPPORT_SUMMARY

Pro následující zpracování dat byly vybrány určité atributy s vhodnou informativní hodnotou z tabulky FACT_AM_SUPPORT_SUMMARY. Jedná se o atributy:

- CUSTOMER_NAME
- INCIDENT_ID
- INCIDENT_TYPE
- INCIDENT_SUB_TYPE
- INCIDENT_DESCRIPTION
- SOURCE
- WASREACTEDTIMESTAMP
- WASSOLVED

5.7.1 Popis atributů

- CUSTOMER_NAME

Záznam obsahující informaci u jakého zákazníka byl generován.

- INCIDENT_ID

Jednoznačný identifikátor každého vygenerovaného záznamu od zákazníka.

- INCIDENT_TYPE

Identifikuje typ záznamu, za jakým účelem byl záznam generován (problém, oznámení, upozornění…)

- INCIDENT_SUB_TYPE

Podskupina pro identifikaci záznamu. Slouží pro případné upřesnění generovaného záznamu.

- INCIDENT_DESCRIPTION

Nejdůležitější část pro každý záznam nesoucí informativní hodnotu pro určitý problém. Tento problém je popsán v textové formě, tedy nestrukturované.

- SOURCE

Popisuje zdroj, na kterém byl záznam generován.

- WASREACTEDTIMESTAMP

Časový záznam, udávající datum a čas kdy přišel na středisko podpory služeb. Od této doby se považoval záznam ve stádiu zpracování a čekaní na vyřešení.

- WASSOLVED

Časový záznam, udávající datum a čas kdy byl vygenerovaný záznam vyřešen. Samotná délka řešení záznamu je tedy rozdílem atributů WASREACTEDTIMESTAMP a WASSOLVED.

Struktura pro tabulku s vybranými atributy pro 10 zákazníků je uvedena v Tabulka 3 - Tvorba tabulky. V tomto případě se neuvádí cesta ke zdrojovému souboru, protože zdroj těchto dat budeme vybírat z již existující tabulky FACT_AM_SUPPORT_SUMMARY.

```
CREATE EXTERNAL TABLE Customers_Top_Ten
(
Customer_name              STRING,     WASSOLVED                TIMESTAMP,
INCIDENT_ID                STRING,     solved_bigint            BIGINT,
INCIDENT_TYPE              STRING,     WASREACTEDTIMESTAMP      TIMESTAMP,
INCIDENT_SUB_TYPE          STRING,     reacted_bigint           BIGINT,
INCIDENT_Description       STRING,     DURATION                 BIGINT,
Source                     STRING,     solved_bool              BOOLEAN
);
```

Tabulka 3 - Tvorba tabulky

5.8 Sledování úloh

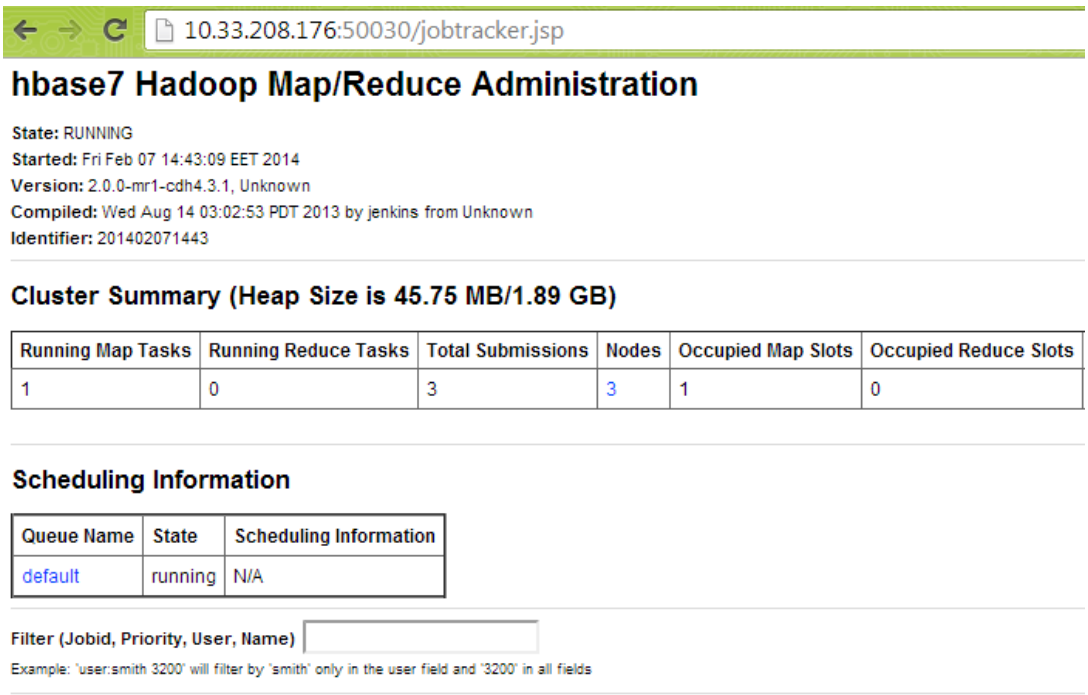
Při jakémkoli příkazu, který je v Hadoop clusteru spojen s prací nad daty je spouštěn JobTracker a TaskTracker, které řídí procesy v HDFS. Obě služby lze z webového prostředí internetového prohlížeče sledovat na příslušných adresách a portech, na kterých jsou spuštěné.

JobTracker (Obrázek 14 - JobTracker), který řídí celý proces komunikace, byl dostupný na adrese http://10.33.208.176:50030/ a TaskTracker na adrese http://10.33.208.176:50070/. Při průběžné aktualizaci stránky při spuštěném úkolu se může kontrolovat stav, ve které části se zrovna zadaný úkol nachází. Zaznamenávají se zde také veškeré naplánované, úspěšně ukončené či neukončené úkoly, které skončily chybou nebo přerušením ze strany uživatele.

5.8.1 JobTracker

V přehledu pro JobTracker se nachází obecné informace o Hadoop clusteru. Je možno přepínat či zobrazovat různé potřebné informace, které jsou součástí nainstalované verze Hadoopu. Shrnuje veškeré

informace ohledně velikosti a naplnění jednotlivých Nodů, které jsou připojeny v clusteru. Dále zobrazuje, které z nich jsou živé a vykazují tedy plnou funkčnost, či mrtvé, tudíž byly zařazeny do černé listiny vyřazených strojů. Veškeré procesy, které jsou nebo byly zadány na vypracování, se automaticky logují a zpětně je uživatel může procházet či filtrovat ve vyhledávači.



Obrázek 14 - JobTracker

5.9 Zjišťování N-gramů

K získání nějaké užitné hodnoty z generovaných nestrukturovaných logů či eventů, je možno využít metodu n-gramů. Analyzováním nejpoužívanějších n-gramů neboli sousloví, lze generovaný záznam kategorizovat do určité skupiny. Množství potřebných n-gramů se odvíjí od různorodosti zdroje. Pokud by se omezil počet n-gramů pouze pro pár nejčastějších, tak se může stát, že velká část záznamů nebude spadat do žádné kategorie, protože jejich textová část nebude obsahovat žádný z nejčastěji používaných n-gramů. Vzhledem k různorodosti obsahové části logů, obzvláště takových, které popisují svůj záznam netypickým nebo příliš krátkým vyjádřením, se stane nemožným získat nějaké opakující se pravidlo v podobě n-gramu pro následnou kategorizaci do nějaké skupiny.

N-gramy lze zjistit v Hivu pomocí HiveQL SELECT příkazu *ngrams (*Ohraničení 1- Zjišťování nGramů*)*. Takovéto vygenerované n-gramy jsou obohaceny o další informace ve strukturované formě společně s počtem výskytů každého n-gramu.

```
SELECT explode( ngrams( sentences( lower( incident_description)), 2, 432)) as ngrams FROM customers_top_ten where source = 'Monitor';
```

Ohraničení 1- Zjišťování nGramů

Pseudokódem popsané zpracování pomocí tohoto příkazu začíná rozsekáním slov do polí. Vznikne dvoudimenzní pole, ve kterém se vyskytují samostatná slova v jedné dimenzi a celek textové části v druhé dimenzi. Funkce ngram poté propočítá frekvenci slov, která se v poli vyskytují, podle zadaných kritérií v parametrech funkce. V mém případě se zaměřuje na vyhledání 432 nejčastějších dvou po sobě jdoucích slovech, z důvodu následné kategorizace.

Aby se na výstupu zobrazily pouze samotné n-gramy s počtem výskytů, je nutno provést formátovací příkaz pro čtení pouze obsahové části uvnitř hranatých a složených závorek. Možné řešení je zobrazeno v Ohraničení 2 - Formátování nGramů.

```
Replace( PurgeChar( SubField( c1, ':'), '[]{}"'), 'estfrequency','') as cleanedNgrams
```

Ohraničení 2 - Formátování nGramů

Z vhodně zvoleného vzorku zjištěných n-gramů se vytvoří list pro určení kategorizace. Pro program QlikView se při importu dat kontroluje obsah textové části záznamů a porovnává jej s listem vytvořených pravidel. Pomocí oddělovače „|“ můžeme určit kategorii, podkategorii, atd. pro vyhledaný n-gram. Jestliže se kategorie nebo podkategorie neurčí, tak záznamy, které obsahují nekategorizovaný n-gram nebudou přiřazeny do žádné kategorie. Výpis části kategorizace pro n-gramy je zobrazen v následující tabulce (Tabulka 4 - Kategorizace).

```
[Links]:
LOAD * inline [
cleanedNgrams                    |  category    |  subcategory
citrixsecuregateway,id,          |  Application |  Citrix          |
is,critical,                     |  Alert       |  Critical        |
not,operational,                 |  Alert       |  Not ok          |
changed,state,                   |  Alert       |  State           |
warning,state.parameter,         |  Alert       |  Warning         |
alarm,of,                        |  Alert       |                  |
boexi40siakesbitest1,is,         |  Application |  BusinessObjects |
service,citrixhostservice,       |  Application |  Citrix          |
backup,device,                   |  Backup      |  Device          |
reply,device,                    |  Device      |                  |
space,left,                      |  Filesystem  |  Capacity        |
unreachable,agent,               |  Network     |  Agent           |
connectionstatus,triggered,      |  Network     |  Connection      |
firewall,prevents,               |  Network     |  Firewall        |
ftp,fetcher,                     |  Network     |  FTP             |
…
] (delimiter is '|');
```

Tabulka 4 - Kategorizace

6 Prezentace dat v programu Qlikview

Zpracovaná data z Hadoop platformy jsem zobrazil celkově ve čtyřech tabulích v programu Qlikview. Každá z tabulí má svůj specifický význam v pohledu k nahlíženým datům. Nejprve je ale nutné uskutečnit napojení programu s Hadoop platformou, aby měl co vizualizovat.

6.1 Napojení dat

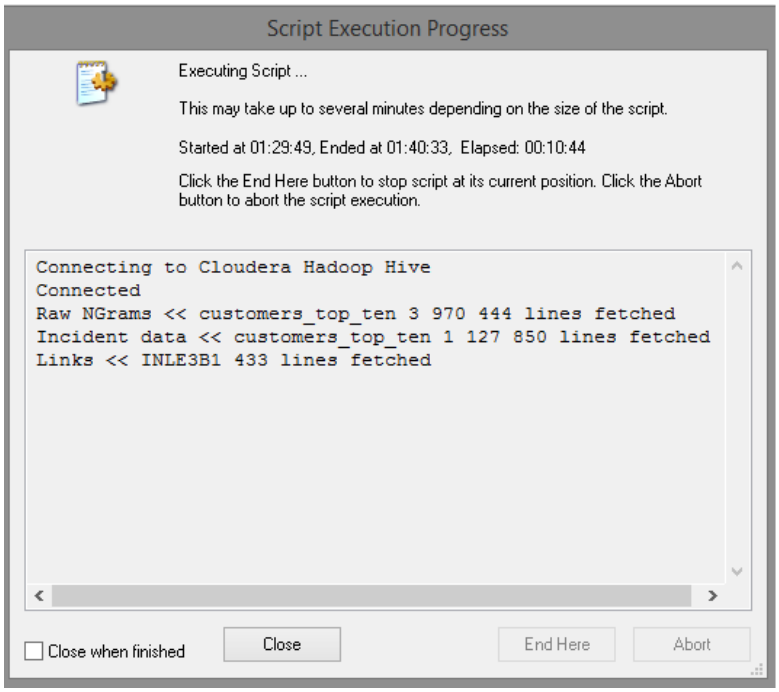
Propojení mezi programem Qlikview a Hadoop clusterem bylo uskutečněno pomocí ODBC konektoru. Ten zajistil přístup do Hadoop clusteru. Nastavuje se ve vlastním prostředí pro program Qlikview. Je potřeba znát IP adresu hostovaného zařízení a port přes jaký má konektor komunikovat. Nativně probíhá komunikace s Hivem přes port 10000. Dále je třeba nastavit na jaký Hive server má přistupovat a autentifikaci přístupu pokud bylo zadáno zabezpečení. V mém případě bylo nutné použít autentifikaci pomocí uživatelského jména.

Pokud konfigurace byla správně nastavena, v editačním skriptu pro nahrávání dat v programu Qlikview bychom již měli mít přístup zvolit vytvořený konektor do Hadoop clusteru.

6.2 Skript zdroje dat

Tento proces nám připojí QV na zdroj dat. Jelikož konfigurace napojení do HDFS k instanci Hivu je již vytvořena, potřebujeme akorát zkonstruovat *connection string* do skriptu. Tento proces zjednoduší nativní funkce po vybrání požadovaného přístupu k databázi, v mém případě přes ODBC protokol. Po vybrání se zvolí datový zdroj a potvrdí se napojení.

Test připojení na databázi k vytvořeným strukturám lze provést pomocí Select builderu v QV. Ten umožní bez znalosti SQL dotazů vytvořit příkaz pro dotaz do potřebné tabulky v databázi. Také vypíše dostupné tabulky, na které se můžeme připojit. Pokud nezobrazí žádnou dostupnou strukturu, tak je špatně provedena konfigurace připojení k databázi, či autorizace.



Obrázek 15 – Kategorizace přes HDFS a načtení dat do QV

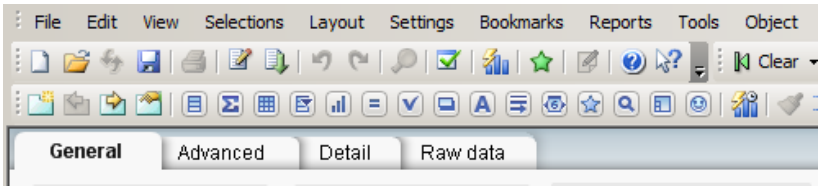
V editačním skriptu jsem vypsal dotaz pro načtení všech záznamů společně s vyhledaným ngramem, ten ovšem nemá výstup jako dvousloví, ale samotný ngram je obsažen v textové zprávě. Je tedy nutno osekat vyhledaný ngram, aby na výstupu bylo zobrazeno samotné dvousloví. Dalším příkazem, který se podle id záznamu napojí na předcházející dotaz, se získají všechny ostatní dostupné atributy. Posled-

ním důležitým příkazem je vytvoření struktury pravidel, podle kterých se budou záznamy kategorizovat podle vyhledaných ngramů. Výpis skriptu je částečně vypsán v Příloze č.2 – Qlikview připojení do HDFS.

Po vytvoření a uložení skriptu je už jen nutné potvrdit načtení dat do QV, která se kompletně stáhnou a není nutné online připojení k databázi. Data jsou dostupná i tehdy, když se připojení s databází ukončí. Proces zjišťování ngramů a jejich kategorizace je poměrně rychlý. V mém případě se zpracovávalo 375 950 záznamů a celý proces trval 10 minut a 44 sekund (viz Obrázek 15 – Kategorizace přes HDFS a načtení dat do QV).

6.3 Vizualizace dat

Zobrazení dat jsem uskutečnil pomocí tabulí neboli záložek. První tabule zobrazuje obecný přehled dat. Základní filtry pro určení odkud data pocházejí, jakého jsou typu, kdy byly vygenerovány a jaký ngram se v daných záznamech identifikoval. V grafu se bude podle zadaných kritérií zobrazovat křivka hodnot, které zobrazují, jak dlouho v určitém časovém období trvalo vyřešit daný problém. Zobrazená hodnota pro jedno časové období je vypočítána jako průměr všech délek trvání záznamu spadající do tohoto časového období.



Obrázek 16 - Záložky v Qlikview

Druhá záložka zobrazuje pokročilý náhled oproti první tabuli. Obsahuje navíc přidruženou kategorii a subkategorii, které byly podle identifikovaného ngramu přidruženy každému záznamu. V grafu je dále zobrazen navíc počet záznamů zpracovávaných v určitém časovém období.

Třetí záložka obsahuje statistický souhrn veškerých použitých pravidel. Ke každému vytvořenému pravidlu je zobrazen počet záznamů spadající do dané kategorie a jejich celkový poměr k použitému vzorku dat. Obrázek této záložky je zobrazen v Příloha č.4 – Qlikview prezentace dat.

Na poslední, čtvrté, záložce jsou zobrazeny v původní podobě bez kategorizace. Tak jak byla načtena ze zdrojové databáze. Jejich nestrukturovaná forma se vyskytuje ve sloupečku „description“, podle které vznikala veškerá kategorizace pro reporting.

7 Srovnání s SQL řešením

Abychom měli srovnání, jestli Hadoop platforma poskytuje výhodu ke snadnému zpracování nestrukturovaných dat, provedl jsem jedno z možných řešení na klasické SQL databázi. Data se zpracovávala na stejném zařízení jako v případě zpracování na Hadoop platformě. Pro řešení jsem tedy vytvořil lokální databázi v Microsoft SQL Management Studiu, ve které jsem vytvořil tabulku a naplnil ji zdrojovými daty. Pravidla, podle kterých chceme záznamy kategorizovat, jsem použil z předešlého zpracování, ať dojdeme ideálně ke stejným výsledkům.

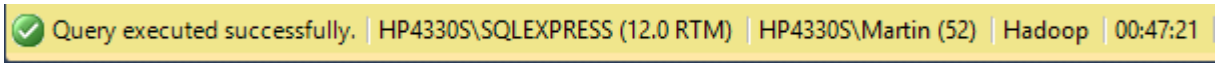
7.1 Kategorizace

Data jsem se rozhodl kategorizovat pomocí procedury, ve které jsem připravil pravidla, podle kterých se vytvořila kopie záznamu s jeho id a přidruženou kategorií a podkategorií. Tohoto principu bylo dosaženo použitím *INSERT* příkazu, který vytvořil záznamy do druhé tabulky použitím *SELECT* dotazu na zdrojovou tabulku. Pro každý typ kategorie byl proveden samostatný dotaz s použitým příkazem *LIKE*, uvedeného v Ohraničení 3 - Příkaz LIKE. Pokud dotaz vrátil záznamy, tak je následně uložil do výsledné tabulky s kategorií, která byla přidružena hledaným ngramům.

```
LIKE  '%ngram1 ngram2%'
```

Ohraničení 3 - Příkaz LIKE

Projít a kategorizovat 400 000 záznamů trvalo SQL databázi necelou hodinu výpočetního času (viz Obrázek 17 - Kategorizace dat pomocí SQL). Databáze zpracovávala celkem 432 kategorií, a jelikož jednu kategorii obsluhuje jeden SQL dotaz, bylo celkově zpracováváno 432 dotazů. Čas potřebný k projití databázové tabulky s daným ngramem pro jeden dotaz trval okolo 7 sekund. Což je opravdu



Obrázek 17 - Kategorizace dat pomocí SQL

vysoká hodnota na relativně malém vzorku dat. Kategorizování dat tímto způsob je tedy špatným řešením a nedá se moc srovnávat v řešením na Hadoop platformě, protože způsob zpracování dat prochází jinými procesy. Pseudokód SQL řešení se totiž výrazně liší od zpracování pomocí Hadoop platformy. V tomto případě se text nijak nerozsekává na menší celky a zpracovává se jako celek a indexace, k efektivnějšímu průchodu tabulkou, lze využít pouze pro část textu před prvním použitým procentem v příkazu *LIKE*. Jelikož v mém případě příkaz *LIKE* začíná procentem, tak není možno využít výhody rychlejšího průchodu tabulkou.

Po vykonání předchozího úkolu je ještě nutné provést kontrolu nekategorizovaných záznamů. Takové záznamy by se nenakopírovaly do výsledné tabulky a mohly by zkreslit konečné statistické hodnoty. Vhodné je například použití SQL příkazu *MERGE*, který porovná záznamy podle zadaného kritéria ze zdrojové a cílové tabulky a pokud hodnotu v cílové tabulce nenalezne, tak id záznamu překopíruje s prázdnou kategorií.

7.2 Prezentace dat

Pro porovnání hodnot zpracovaných na SQL platformě jsem použil stejný prezentační dokument jako v případě dat zpracovaných na platformě Hadoop. Pozměnil jsem pouze přístup ke zdrojovým datům a jednotlivé grafy a statistické tabulky se tedy řídí stejnými pravidly.

7.3 Alternativní řešení na SQL platformě

Abychom mohli SQL a Hadoop platformu lépe porovnat mezi sebou, jak jsou tyto dvě platformy schopné zpracovávat nestrukturovaná data, musíme vytvořit jiný druh zpracování dat na SQL platformě. Obsah textové části by bylo nutné předzpracovat, tedy rozdělit text na dvouslovné ngramy. Následně vytvořit indexaci na rozsekaných částech textu, čímž se výrazně zefektivní průchod tabulkou při vyhledávání ngramů a odpadne nám z dotazu použití příkazu *LIKE*. Tento proces by tak měl výrazně urychlit vyhledávání ngramů v tabulce, jejich kategorizaci a napodobit způsob zpracování dat, kterého bylo dosaženo na Hadoop platformě.

8 Závěr

Cílem práce bylo zpracovat nestrukturovaný obsah záznamů, které byly generovány různými stroji či lidmi na podpoře služeb. Úkolem bylo zhodnotit, jaký obsah zpráv se zpravidla generuje a pomocí ngramů tyto zprávy vhodně kategorizovat. Platforma pro zpracování těchto dat byla distribuovaný souborový systém od společnosti Apache Hadoop. Jelikož počet dat nebyl nijak velký, necelých 400 000 záznamů, bylo potřeba také otestovat, jaký přínos tato platforma vzhledem ke klasickým databázovým systémům nabízí při zpracování neupravených nestrukturovaných obsahů logů.

V první části, teoretické, této práce jsem se zaměřil na seznámení s konceptem Big Data a proč je v této době neopominutelným prvkem. Popsal jsem důvody počátku vzniku distribuovaného zpracovávání dat a firem, které se touto problematikou zabývaly.

Dále jsem rozebral distribuovaný souborový systém společnosti Apache. V čem se toto řešení liší od klasických databází a obecnou specifikaci komunikace uvnitř této platformy. Popsal jsem způsob paralelního zpracování dat pomocí metody map-reduce a její přínos v distribuovaném souborovém systému.

V praktické části své práce jsem zprovoznil Hadoop platformu na vlastním počítači pomocí virtuálního prostředí od společnosti Cloudera, na kterém jsem tuto práci zpracovával. Pro připojení do HDFS k datům z lokálního stroje do virtuálního prostředí obstarával ODBC konektor pro Hive.

Výsledná vizualizace zpracovaných dat proběhla v programu Qlikview, ve kterém byly vytvořeny pravidla kategorizace pro nestrukturovaný obsah záznamů. Data byla vyobrazena na několika záložkách s odlišnými statistikami. Kategorizovány byly zhruba dvě třetiny záznamů, jelikož se v nich nacházely podobné části struktur, pomocí nichž je bylo možno přiřadit dané kategorii. Zbytek nekategorizovaných záznamů měl ve své obsahové části velice málo shodných ngramů s ostatními záznamy nebo jeho obsah neměl žádnou vypovídající hodnotu.

Pro zjištění přínosu Hadoop platformy pro zpracování nestrukturovaných dat jsem provedl řešení pomocí SQL databáze. Vytvořenou tabulku naplněnou zdrojovými daty jsem zpracovával pomocí procedury, která obsahovala sadu dotazů pro vyhledávání ngramů, tedy pravidel stejných jako v případě zpracování na Hadoop platformě, a zjišťoval, zda obsahují daný ngram. Pokud ano, tak jsem tento záznam s kategorií nakopíroval do druhé tabulky. Řešení se ukázalo jako naprosto nevhodné pro zpracování předem neupravených dat. Průchod tabulkou a vyhledání ngramu trvalo pro jeden dotaz přibližně 7 sekund a celý proces kategorizace zabral necelých 50 minut. Text by bylo nutné předem zpracovat a rozdělit na potřebné ngramy, aby je mohla SQL platforma efektivněji procházet a kategorizovat.

Kategorizace dat nám nakonec ukázala, jaké záznamy se na podporu služeb nejčastěji zasílají, díky přiřazení kategorií jednotlivým záznamům. K těmto kategoriím byl také přidružen čas, potřebný k vyřešení jednotlivých vygenerovaných problémů v rámci kategorie, do které byly záznamy přidruženy. Tyto informace by mohly být následně využity k zefektivnění chodu podpory služeb, pokud by délka zpracovávání problému z nějaké kategorie trvala výrazněji déle než u jiných. Případně k prozkoumání důvodů, proč k této odchylce došlo.

9 Použitá literatura

9.1 Terminologický slovník

Android	operační systém společnosti Google pro chytrá zařízení
Big Data	pojem pro vysoce objemná či kvantitativní data
byte	jednotka pro měření velikosti dat
DataNode	datový uzel
Electronic Data Interchange	standard pro výměnu zpráv mezi komunikujícími stanicemi
email	elektronická pošta
Extensible Markup Language	obecný značkovací jazyk
Facebook	sociální síť
Google File System	Google souborový systém
Hadoop Distributed File System	Hadoop distribuční souborový systém
Handshake	navázání spojení přes TCP protokol
Heartbeat	zpráva DataNodu o své aktivitě
JobTracker	sledovač prací v HDFS
MapReduce	vzor pro paralelní zpracování velkých objemů dat
master/slave	Princip nadřízeného a podřízeného
Metadata	data o datech
NameNode	hlavní (primární) uzel
Nutch Distributed File System	Nutch distribuční souborový systém
Random-access memory	paměť s přímým přístupem nebo libovolným výběrem
Secondary NameNode	záloha hlavního uzlu
status	příspěvek uživatele sociální sítě Facebook
TaskTracker	sledovač úkolů v HDFS
tweet	příspěvek uživatele sociální sítě Twitter
Twitter	sociální síť
Windows Phone	operační systém společnosti Microsoft pro mobilní zařízení

9.2 Seznam použitých symbolů a zkratek

BD	Big Data
EDI	Electronic Data Interchange
GB	Gigabyte, 10^9 bytů
GFS	Google file system
HDFS	Hadoop Distributed File System
M/R	MapReduce
MB	Megabyte, 10^6 bytů
QV	Qlikview
RAM	Random-access memory
RF	replikační faktor
TB	Terabyte, 10^{12} bytů
TCP	Transmission Control Protocol
XML	Extensible Markup Language

9.3 Zdroje

1. **Palfreyman, John.** Insight on Business. *Big Data - Vexed by Veracity.* [Online] 20. 5 2013. [Citace: 5. 1 2014.] http://insights-on-business.com/government/big-data-vexed-by-veracity/.

2. **Kelly, Jeff.** Wikibon. *Big Data in the Aviation Industry.* [Online] 16. 9 2013. [Citace: 6. 1 2014.] http://wikibon.org/wiki/v/Big_Data_in_the_Aviation_Industry.

3. **Alfa.cz.** Alfa Computer a.s. *Interní 3,5" SATA Disky.* [Online] 2014. [Citace: 29. 1 2014.] http://www.alfacomp.cz/php/index.php?eid=15L14007H1TZ.

4. **Tech Tracker.** Storage: From Highway Robbery to Runaway Bargain. *PC Magazine.* 26, 2007, Sv. 19, October 2.

5. **Hill, David.** Network Computing. *The Three Transformation of IT.* [Online] 3. 5 2013. [Citace: 28. 12 2013.] http://www.networkcomputing.com/storage-networking-management/the-three-transformations-of-it/240154113.

6. **Jannsen, Cory.** techopedia. *Semi-Structured Data.* [Online] 2014. [Citace: 30. 1 2014.] http://www.techopedia.com/definition/28802/semi-structured-data.

7. **Chhibber, Varun.** Infosys. *Butterfly Effect: Analytics and Social media.* [Online] 2. 7 2012. [Citace: 24. 1 2014.] http://www.infosysblogs.com/thought-floor/2012/07/butterfly_effect_analytics_and.html.

8. **Gantz, John a Reinsel, David.** *Extracting Value from Chaos.* [pdf] Spojené státy americké : IDC iview, 2011.

9. **Wood, Tim a Chernev, Iskren.** WIPRO. *Big Data.* [Online] 2014. [Citace: 16. 1 2014.] http://www.wipro.com/landing-pages/infographic.aspx.

10. **White, Tom.** *Hadoop: The Definitive Guide.* edice druhá. Spojené státy americké : O'Reilly Media, 2011. 978-1-449-38973-4.

11. **Alfa computer a.s.** Alfa.cz. *Interni SSD SATA 2,5" disky.* [Online] 1 2014. [Citace: 30. 1 2014.] http://www.alfacomp.cz/php/index.php?eid=15L14007H2AN.

12. **Karmasphere.** Karmasphere. *Deriving intelligence from big data in hadoop.* [Online] 21. 9 2011. [Citace: 22. 12 2013.] http://www.karmasphere.com/Resource-Center/deriving-intelligence-from-big-data-in-hadoop.html.

13. **Rouse, Margaret.** Search Business Analytics. *Hadoop cluster.* [Online] 6 2013. [Citace: 16. 1 2014.] http://searchbusinessanalytics.techtarget.com/definition/Hadoop-cluster.

14. **Borthakur, Dhruba.** Apache. *HDFS Architecture Guide.* [Online] 8. 4 2013. [Citace: 1. 22 2014.] http://hadoop.apache.org/docs/stable1/hdfs_design.html.

15. **Manikanta, Yaswanth.** *Heart of Hadoop.... HDFS.* [Online] 5. 1 2014. [Citace: 21. 1 2014.] http://hadoopfox.blogspot.cz/2014/01/heart-of-hadoop-hdfs.html.

16. **Hedlund, Brad.** Brad Hedlund. *Understanding Hadoop Clusters and the Network.* [Online] 10. 9 2013. [Citace: 20. 1 2014.] http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/.

17. **-.** Hadoop Wiki. *JobTracker.* [Online] 30. 6 2010. [Citace: 16. 1 2014.] http://wiki.apache.org/hadoop.

18. **Mallassi, Olivier.** Octo. *How to "crunch" your data stored in HDFS?* [Online] 27. 10 2010. [Citace: 21. 1 2014.] http://blog.octo.com/en/how-to-crunch-your-data-stored-in-hdfs/.

19. **Rouse, Margaret.** SearchCloudComputing. *MapReduce.* [Online] 8. 2 2010. [Citace: 24. 1 2014.] http://searchcloudcomputing.techtarget.com/definition/MapReduce.

20. **Schulte, Garth.** Apache Hadoop. *CBT Nuggets.* [Online] 9. 3 2013. [Citace: 6. 2 2014.] https://www.cbtnuggets.com/it-training-videos/course/cbtn_hadoop.

21. **Powell, Travis.** Statistics And DataMining. *Apache wiki.* [Online] 8. 8 2011. [Citace: 1. 4 2014.]

22. **Tieto.** About Us. [Online] [Citace: 3. 3 2014.] http://www.tieto.com/about-us.

23. **Zikopoulos, Paul C., a další.** *Harness the Power of Big Data: The IBM Big Data Platform.* Spojené státy americké : The McGrow-Hill Companies, 2013. 978-0-07-180817-0.

9.4 Seznam obrázků

9.5 Seznam příloh

Příloha č.1 – Tvorba struktury v HDFS

Příloha č.2 – Qlikview připojení do HDFS

Příloha č.3 – SQL kategorizace

Příloha č.4 – Qlickview prezentace dat

Příloha č.1 – Tvorba struktury v HDFS

```
hadoop fs -put customers_top_ten.csv /user/cloudera/customers_top_ten.csv;
```

```
CREATE EXTERNAL TABLE customers_top_ten
(
customer_name STRING,
incident_id STRING,
incident_description STRING,
incident_type STRING,
incident_sub_type STRING,
source STRING,
wasreactedtimestamp timestamp,
wassolved timestamp,
duration bigint,
ReactedTimeInSec bigint,
SolvedTimeInSec bigint,
solved_bool boolean
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "\;"
LOCATION '/user/cloudera/customers_top_ten';
```

```
LOAD DATA inpath '/user/cloudera/customers_top_ten.csv' INTO TABLE customers_top_ten;
```

Příloha č.2 – Qlikview připojení do HDFS

```
ODBC CONNECT TO [Cloudera Hadoop Hive];


[Raw NGrams]:

Load incident_id, _c1 as ngrams, Replace(PurgeChar(SubField(_c1,':'), '[]{}"'), 'estfrequency','') as cleanedNgrams;

SQL select incident_id, ngrams(sentences(lower(incident_description)), 2, 5) from customers_top_ten group by incident_id;


[Incident data]:

SQL select * from customers_top_ten;



[Links]:

LOAD * inline [

cleanedNgrams|category|subcategory

citrixsecuregateway,id,|Application|Citrix|

is,critical,|Alert|Critical|

not,operational,|Alert|Not ok|

changed,state,|Alert|State|

warning,state.parameter,|Alert|Warning|

alarm,of,|Alert||

…

] (delimiter is '|');
```

Příloha č.3 – SQL kategorizace

```
CREATE PROCEDURE [dbo].[Categorization]

AS

BEGIN

INSERT INTO CategorizedDataS ( incident_id, Category, SubCategory) SELECT incident_id, 'Application', 'Citrix' FROM RawDataU WHERE incident_description LIKE '%citrixsecuregateway id%';

INSERT INTO CategorizedDataS ( incident_id, Category, SubCategory) SELECT incident_id, 'Alert', 'Critical' FROM RawDataU WHERE incident_description LIKE '%is critical%';

INSERT INTO CategorizedDataS ( incident_id, Category, SubCategory) SELECT incident_id, 'Alert', 'Not ok' FROM RawDataU WHERE incident_description LIKE '%not operational%';

INSERT INTO CategorizedDataS ( incident_id, Category, SubCategory) SELECT incident_id, 'Alert', 'State' FROM RawDataU WHERE incident_description LIKE '%changed state%';

INSERT INTO CategorizedDataS ( incident_id, Category, SubCategory) SELECT incident_id, 'Alert', 'Warning' FROM RawDataU WHERE incident_description LIKE '%warning state.parameter%';

…

merge CategorizedData c

USING RawData r

on c.incident_id = r.incident_id

WHEN NOT MATCHED THEN

INSERT (

customer_name, incident_id, incident_description, incident_type, incident_sub_type,

source, wasreactedtimestamp, wassolved, duration, ReactedTimeInSec, SolvedTimeInSec, SolvedBool, Category, SubCategory)

VALUES (

r.customer_name, r.incident_id, r.incident_description, r.incident_type, r.incident_sub_type,

r.source, r.wasreactedtimestamp, r.wassolved, r.duration, r.ReactedTimeInSec, r.SolvedTimeInSec, r.SolvedBool, '-', '-' );

END;
```

Příloha č.4 – Qlickview prezentace dat

No of incidents per category

No of incidents per category

Category
-
Alert
Application
Backup
Check
Critical
Database
Device
Filesystem

No of incidents per subcategory

No of incidents per subcategory

Subcategory
-
Agent
AIX
Authentication
Backup
BusinessObjects
Capacity
Citrix
Cognos
Connection

Current Selections

Incident Category Statistics

Category	Subcategory	No of incidents	No of incidents in %	Average duration of solved inciden...
		375 950	100,00%	11,23
Filesystem	Capacity	368	0,05%	0,87
Critical		1 672	0,22%	1,12
Network	Timeout	1 672	0,22%	1,12
Network	sysmon	1 556	0,20%	1,33
Alert	State	1 562	0,20%	1,63
SAP	Backup	375	0,05%	2,14
System health		546	0,07%	2,23
Application	Patrol Agent	57 777	7,51%	2,67
	Agent	4 490	0,58%	2,71
RTCloud		6 286	0,82%	3,38
System health	CPU	18 249	2,37%	3,48
Process		5 006	0,65%	3,51
Backup	MSSQL	17 698	2,30%	3,81
Backup	Device	1	0,00%	3,88
SAP	Log	764	0,10%	4,10
Security	AIX	820	0,11%	4,20
Network	Connection	31 500	4,09%	4,26
Application	Domino	1	0,00%	4,28
Network	Node Down	3 145	0,41%	4,55
Network		32 652	4,24%	4,63
Application	Cognos	140	0,02%	5,00
SAP	NetWeaver	5 292	0,69%	5,13
SAP		12 488	1,62%	5,23
Test		8 872	1,15%	5,43
Network	Unreachable	3 771	0,49%	5,57
SAP	Oracle	2 883	0,37%	6,00
Device	IOS	3 393	0,44%	6,10
Alert	Warning	1 809	0,24%	6,60
System health	UNIX	2	0,00%	7,00
Network	No response	9 065	1,18%	7,31
Database		30	0,00%	7,53
Network	Firewall	935	0,12%	7,65
Network	FTP	4 340	0,56%	7,76
Application	Citrix	7 673	1,00%	8,23
System health	utilization	42 016	5,46%	8,95
Backup		25 886	3,36%	8,96
Network	NIC	12 628	1,64%	9,05